CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Masterstudiengänge Informatik & Wirtschaftsinformatik

Projektgruppe: Dynamische Portfoliooptimierung eines virtuellen Kraftwerks

Dokumentation

vorgelegt von

Stephan Balduin
Dierk Brauer
Lars Elend
Stefanie Holly
Jan Korte
Carsten Krüger
Almuth Meier
Frauke Oest
Immo Sanders-Sjuts
Torben Sauer
Marco Schnieders
Robert Zilke

Gutachter:

Prof. Dr. Michael Sonnenschein
Dr. Ute Vogel
Dr. Christian Hinrichs

Oldenburg, 1. April 2016

Vorwort

Dieser Abschlussbericht ist das Ergebnis der Projektgruppe *Dynamische Portfoliooptimierung eines virtuellen Kraftwerks* an der Carl von Ossietzky Universität Oldenburg, Department für Informatik, Abteilung Umweltinformatik.
Die Projektgruppe besteht aus zwölf Personen, begann im April 2015 und wurde im März 2016 abgeschlossen.
Der Name des entwickelten Systems lautet *Profit Optimization With Distributed Energy Resources* (*Powder*).

Abstract

The aggregation of distributed energy resources in virtual power plants is a necessary action to overcome barriers to market entry and therefore be able to trade the generated energy in power exchange markets such as the EPEX Spot. The coordination of the various generation units and the selection of the market products and their quantity are two complex interconnected optimization problems. The objective of this project is to automate the optimization of a composition, consisting of the product portfolio and the matching operation schedule to be determined for the virtual power plant, in terms of profit maximization.

Intricate models of generation units are implemented in order to produce feasible schedules. Furthermore machine learning techniques are used to determine a market forecast. The feasible schedules and the market forecast are used as input for the optimization process. During optimization a combination of heuristic algorithms such as simulated annealing or tabu search and linear optimization is applied.

Zusammenfassung

Kleine dezentrale Energieanlagen werden in Anlagenpools aggregiert und über virtuelle Kraftwerke (VK) gesteuert, um Markteintrittsbarrieren an Energiebörsen wie der EPEX SPOT zu überwinden. Zur Maximierung des Gewinns beim Handel der durch das VK bereitgestellten Energie müssen zwei wechselseitig verbundene Optimierungsprobleme gelöst werden. Zum einen erfolgt eine Auswahl von Marktprodukten inklusive ihres Umfangs und des Preises, zu dem sie angeboten werden (Produktportfoliooptimierung). Zum anderen wird für jede Anlage des VKs ein geeigneter Fahrplan bestimmt (Einsatzplanoptimierung). Die Optimierungen sind voneinander abhängig, da der gewählte Einsatzplan die für den Verkauf verfügbare Energiemenge bestimmt, zugleich aber auch der Einsatzplan auf Grundlage des gewählten Produktportfolios gewählt werden kann. Das Ziel ist, die Gesamtoptimierung zu automatisieren, so dass eine Kombination aus Produktportfolio und Einsatzplan gewählt wird, welche den maximal erreichbaren Gewinn erwirtschaftet.

Im Rahmen der Projektgruppe wurden zur Ermittlung möglicher Anlagenfahrpläne und deren Kosten umfangreiche Anlagenmodelle implementiert. Diese Modelle und mittels maschinellen Lernens prognostizierte Marktpreise gehen in die Optimierung des Produktportfolios und des Einsatzplans ein. Dabei werden verschiedene Heuristiken wie Simulated Annealing und Tabusuche mit linearen Optimierungsverfahren kombiniert. Die Marktprognose sowie die einzelnen Optimierer wurden metaoptimiert und anschließend evaluiert.

Inhaltsverzeichnis

1. Einleitung

Seit den letzten Jahren wird zunehmend Energie aus erneuerbaren Quellen produziert, insbesondere da dies durch staatliche Subventionen nach dem Erneuerbare-Energien-Gesetz (EEG) gefördert wird. Dadurch ist der Anteil der dezentralen Energieanlagen gestiegen, somit aber auch die Ausgaben für die Subventionen. Mit der Erweiterung des EEG von 2012 wurde das Abrechnungsmodell der Direktvermarktung für erneuerbare Energieanlagen eingeführt. Dieses erlaubt den Stromerzeugern einen einfacheren Zugang durch Direktvermarkter an den Strommarkt. Die Direktvermarktung und die Verringerung der Subventionen einiger Anlagentypen sollen helfen, die Ausgaben für Subventionen zu verringern. In einem Bericht des Bundesverband der Energie- und Wasserwirtschaft e.V. (BDEW) wurde die Entwicklung der Direktvermarktung von dem aus erneuerbaren Energien erzeugten Strom in Deutschland betrachtet [BDE13]. Aus der Abbildung 60 (im Anhang C) geht hervor, dass mit einer Steigerung der Direktvermarktung bis 2017 gerechnet wird. Da der Handel am Markt jedoch technische Regularien wie eine Mindestgebotsmenge aufweist, ist es weiterhin auch durch die Direktvermarktung nicht möglich, mit einzelnen Anlagen am Markt zu handeln. Eine gängige Methode, um Markteintrittsbarrieren zu durchbrechen, ist der Zusammenschluss von einzelnen dezentralen Energieerzeugungsanlagen zu einem virtuellen Kraftwerk. Um durch die Direktvermarktung mit den Anlagen eines virtuellen Kraftwerks zu handeln, wird ein Vertrag mit einem Direktvermarkter abgeschlossen. Dieser erstellt ein Produktportfolio, indem er die am meisten gewinnbringenden Produkte auswählt, die leistungstechnisch durch die Anlagen eines virtuellen Kraftwerks abgedeckt werden können. Die Verfügbarkeit der einzelnen Anlagen im VK ist vertraglich zugesichert. Jedoch stellt sich die Frage, ob diese Produktportfolios mit den Portfolios anderer, konventioneller Kraftwerksbetreiber konkurrieren können. Trotzdem sollte die Wirtschaftlichkeit aus Sicht jedes einzelnen Anlagenbetreibers des virtuellen Kraftwerks durch das Produktportfolio immer noch gegeben sein. Diese Frage weist darauf hin, dass es sich bei der Erstellung des Produktportfolios um ein Optimierungsproblem handelt. Durch dessen Lösung soll ein Beitrag zur Konkurrenzfähigkeit der Stromerzeugung aus erneuerbaren Energien am Markt beigesteuert werden.

1.1. Problembeschreibung

Um die Konkurrenzfähigkeit von virtuellen Kraftwerken mit einer hohen Anzahl von dezentralen Energieanlagen im Vergleich zu konventionellen Kraftwerken an der Börse zu steigern, ist es von zentraler Bedeutung, ein möglichst gutes Produktportfolio anzubieten. Dabei müssen alle Marktrichtlinien wie Markteintrittsbarrieren oder die pünktliche Abgabe von Auktionen (Beispiel: Day-Ahead-Handel) eingehalten werden. Weiterhin sollten die Produkte ausgewählt werden, mit denen der größte Gewinn erzielt werden kann. Hierbei muss jedoch beachtet werden, dass die benötigte Auktionsmenge (Leistung) zu jedem Zeitpunkt des gehandelten Zeitraums von den Anlagen bereitgestellt werden kann. Dazu wird neben dem Produktportfolio ein Einsatzplan benötigt, welcher auch an die Börse übergeben wird und sich aus den aggregierten Fahrplänen der Anlagen zusammensetzt. An der Börse kann über den Einsatzplan jedes Kraftwerks eingeschätzt werden, ob das Kraftwerk seine Leistung auch einhält. Je mehr Anlagen in diesem Zusammenhang eingesetzt werden, desto mehr Möglichkeiten gibt es, die benötigte Leistung aufzubringen, so dass ein weiteres Optimierungsproblem entsteht.

Für den optimalen Einsatzplan ist es notwendig, dass Fahrpläne verwendet werden, die auch von den Anlagen umgesetzt werden können. Fast jeder Anlagentyp (Windkraftanlage, Blockheizkraftwerk, usw.) besitzt Restriktionen, aufgrund derer nicht jede Betriebsweise der Anlage möglich ist. Blockheizkraftwerke müssen beispielsweise eine definierte Mindestzeit inaktiv sein, nachdem sie ausgeschaltet wurden. Des Weiteren ist es wichtig, die Kosten für die Erzeugung der Leistung für jeden Fahrplan in Zusammenhang zu betrachten, damit nicht die Fahrpläne ausgewählt werden, bei denen die Einsatzkosten am höchsten sind.

Da bereits bei einer kleinen Anzahl von dezentralen Anlagen die Optimierungsprobleme nicht mehr per Hand lösbar sind, sollte ein automatisiertes System dafür entwickelt werden. Zudem darf auch die Zeitrestriktion für die Abgabe der Kontrakte in den Auktionen nicht außer Acht gelassen werden, d. h. das System muss die rechenaufwendige Optimierung in der verfügbaren Zeit vornehmen. Die meisten Anlagenpools für die Direktvermarktung und den Handel an der Börse werden in virtuellen Kraftwerken zusammengefasst. Deren Betreiber nehmen zumeist auch die Position des Direktvermarkters ein. Daher wäre es sinnvoll, die Berechnung des Produktportfolios und des Einsatzplans in dem virtuellen Kraftwerk durchzuführen oder in einem Tool, welches eine Schnittstelle zu dem virtuellen Kraftwerk hat. Weiterhin ist es erstrebenswert, dass für das Produktportfolio nicht nur der Gewinn, sondern auch die Wahrscheinlichkeit für die Annahme der Kontrakte in den Auktionen betrachtet wird. Dies ist natürlich ausschlaggebend für den Erhalt von Erlösen. Die Umsetzung und Lösung dieser Probleme ist ein wichtiger Schritt, um die Anlagenpools noch effizienter und sicherer an dem Börsenhandel teilnehmen zu lassen. Aus der Problembeschreibung wird im Weiteren eine konkrete Zielstellung entwickelt.

1.2. Ziel der Projektgruppe

Das Ziel der Projektgruppe ist die Entwicklung eines Systems, welches automatisiert das Produktportfolio und den Einsatzplan eines virtuellen Kraftwerks für den Day-Ahead-Handel von Energie am Energiemarkt optimiert. Diese Optimierung hat die Maximierung des Gewinns durch das VK zum Ziel. Dabei müssen die Restriktionen der Energieanlagen und des Marktes berücksichtigt werden. Zusätzlich soll das System bereits in der Praxis eingesetzte Anlagenschnittstellen verwenden, um an reale Anlagen angebunden werden zu können.

1.3. Ziel der Dokumentation

Dieser Projektendbericht soll Informationen und Erfahrungen der Projektgruppe *Dynamische Portfoliooptimierung eines virtuellen Kraftwerks* für mögliche Zielgruppen und Stakeholder vermitteln, sowie darüber hinaus eventuell zu weitergehenden Überlegungen anregen. Grundsätzlich soll diese Dokumentation im Projekt erlangtes Wissen vermitteln.

1.4. Aufbau des Dokuments

Dieser Projektbericht schildert nach der Einleitung zunächst das Vorgehen innerhalb des Projekts aus Sicht des Projektmanagements. Dies umfasst den Projektablauf, das eigens entwickelte Vorgehensmo-

dell, die von den Mitgliedern ausgefüllten Rollen, das Qualitätsmanagement, die Arbeitsinfrastruktur und -organisation, sowie weitere zentrale Aufgaben.

Das Kapitel Vision und Related Work definiert die strategische Ausrichtung des Projekts auf Basis der gegebenen Problembeschreibung und ist zugleich die Grundlage der Motivation für das Projekt. Außerdem werden verwandte wissenschaftliche Arbeiten im Umfeld des Themengebietes auf ihre Relevanz geprüft und zu dieser Arbeit hin abgegrenzt.

Im Kapitel Anforderungen werden die ermittelten Stakeholder vorgestellt und es wird auf die Modellierungsmethode Smart Grid Architecture Model (SGAM) eingegangen, die speziell für den Energiebereich entwickelt wurde. Nach der SGAM-Analysephase und der SGAM-Architekturphase, in denen bereits erste Anforderungen erhoben werden, folgen weitere, spezifische Anforderungen. Die Erfüllung der Anforderungen kann in den Anforderungstabellen in diesem Kapitel eingesehen werden.

Im Kapitel Technologien folgt eine Beschreibung der Technologien, die bei der Entwicklung von *Powder* eingesetzt wurden. Außerdem wird begründet, warum sich die jeweilige Technologie gegenüber ihren Alternativen durchgesetzt hat. Ob sich die eingesetzten Technologien im Projektalltag bewährt haben, wird in einer anschließenden Evaluation reflektiert.

Die grundlegende Systemarchitektur wird im Kapitel Architektur beschrieben. Zunächst wird die grundlegende Struktur vorgestellt und auf die Kommunikationsschnittstellen eingegangen. Anschließend werden Grundlagen, Konzepte und Implementierung der einzelnen Systemkomponenten erläutert. In Abschnitt 6.10 werden außerdem die Evaluationsstrategien für die Evaluation der Optimierer vorgestellt. Am Ende des Kapitels wird die Benutzerschnittstelle des Systems beschrieben.

Im Kapitel Tests wird die Notwendigkeit von Tests für die Verifikation der Funktionalitäten erläutert. Außerdem werden Technologien beschrieben, die bei der Umsetzung der Testfälle eingesetzt wurden.

Im Fazit wird abschließend das Projekt reflektiert und ein Ausblick gegeben.

Dem Endbericht werden zuletzt jene Dokumente angehängt, die nicht direkt der Projektarbeit zuzuordnen sind oder den Lesefluss, beispielsweise aufgrund des Umfangs, stören würden. Gesondert nennenswert sind dabei die erarbeiteten Inhalte der vor dem Projektstart abgehaltenen Seminare in Form eines Seminarbands. Da in wissenschaftlichen Arbeiten eine geschlechtergerechte Sprache zu verwenden ist, wurden verschiedene Varianten betrachtet und entschieden, in diesem Bericht die männliche Schreibform anzuwenden. Dies geschah unter Zustimmung aller Projektteilnehmer.

2. Projektmanagement

Nach [BS] ist das Projektmanagement dafür zuständig, die Erwartungen der Stakeholder zu erfüllen. Zunächst müssen die Interessen der Stakeholder transparent gemacht werden. Anschließend wird das Projekt in Abstimmung mit den Stakeholdern geplant und die Interessen werden priorisiert. Elementarer Bestandteil des Projektmanagements ist, die richtige Vorgehensweise zur Durchführung eines Projekts zu finden. Die Wahl der Vorgehensweise richtet sich in der Regel nach folgenden Kriterien:

- Umfang
- Komplexität
- Umfeld bzw. Branche
- Projektart (z. B. Entwicklungsprojekt, Lernprojekt oder Wartungsprojekt)
- Vorgaben des Auftraggebers.

Zu den Aufgaben gehört es außerdem, geeignete Projektmanagementwerkzeuge auszuwählen und zur Anwendung zu bringen. Dazu zählen beispielsweise Software-Werkzeuge oder Interview-Methoden. Diese Entscheidung sollte frühestmöglich, also vor Projektbeginn, getroffen werden. Die Projektgruppe virtuelles Kraftwerk konnte sich größtenteils an diese Grundlagen halten, lediglich Software-Werkzeuge wurden zum Teil erst mit Beginn der Implementierung evaluiert.

Die weiteren Abschnitte beschreiben detailliert einige Projektmanagement-Vorgänge in der Projektgruppe. Das Kapitel Projektablauf (s. Abschnitt 2.2) gibt einen Überblick über einige Abläufe innerhalb des Projekts, wie der Sitzungen oder des Vorgehensmodells des Projekts. Das Kapitel Vorgehensmodell (s. Abschnitt 2.1) beschreibt anschließend ausführlich das eigens entwickelte Modell ScrUP. Das Kapitel Rollen (s. Abschnitt 2.3) beschreibt die eingeführten Rollen, die von der Projektgruppe als wichtig erachtet wurden. Im Kapitel Qualitätsmanagement (s. Abschnitt 2.4) wird erläutert, inwiefern Qualitätsmanagement im Projekt relevant ist. Im Kapitel Infrastruktur (s. Abschnitt 2.5) werden abschließend Werkzeuge erläutert, mit deren Hilfe Projektarbeit strukturiert und erleichtert werden kann.

2.1. Vorgehensmodell

Zu Beginn des Projekts wurde von der Projektgruppe ein Vorgehensmodell entwickelt, nach dessen Muster die Arbeitsabläufe im Projekt abgehandelt werden sollten. Dieser zusätzliche Aufwand wurde getätigt, um Schwierigkeiten, basierend auf Erfahrungen aus vorherigen akademischen Studierendenprojekten, vorzubeugen bzw. auf diese einzugehen. Ein fester Abschlusstermin und die voraussehbare Tatsache, dass Gruppenmitglieder zeitweise mehr oder weniger zur Verfügung stehen, führen zudem zu Planungsherausforderungen. Das hier vorgestellte Vorgehensmodell basiert grundlegend auf Scrum, wie es beispielhaft von Boris Gloger in [Glo13] beschrieben wurde. Eine genaue Erklärung von Scrum kann auch in Anhang Kapitel 11 im Anhang G nachgelesen werden. Zusätzlich wurden Aspekte des Unified Process für das Zeitmanagement und die *Inception*- sowie *Elaboration*-Phase

aufgegriffen. Auf diese Weise ergibt sich das Vorgehensmodell Scrum Unified Process (ScrUP), dessen Name ein Akronym beider Modelle ist und sich aus Teilen beider Modelle zusammensetzt. Es wird in Abbildung 1 und Abbildung 2 dargestellt. Anschließend werden einige Schlüsselpunkte des Modells vorgestellt.

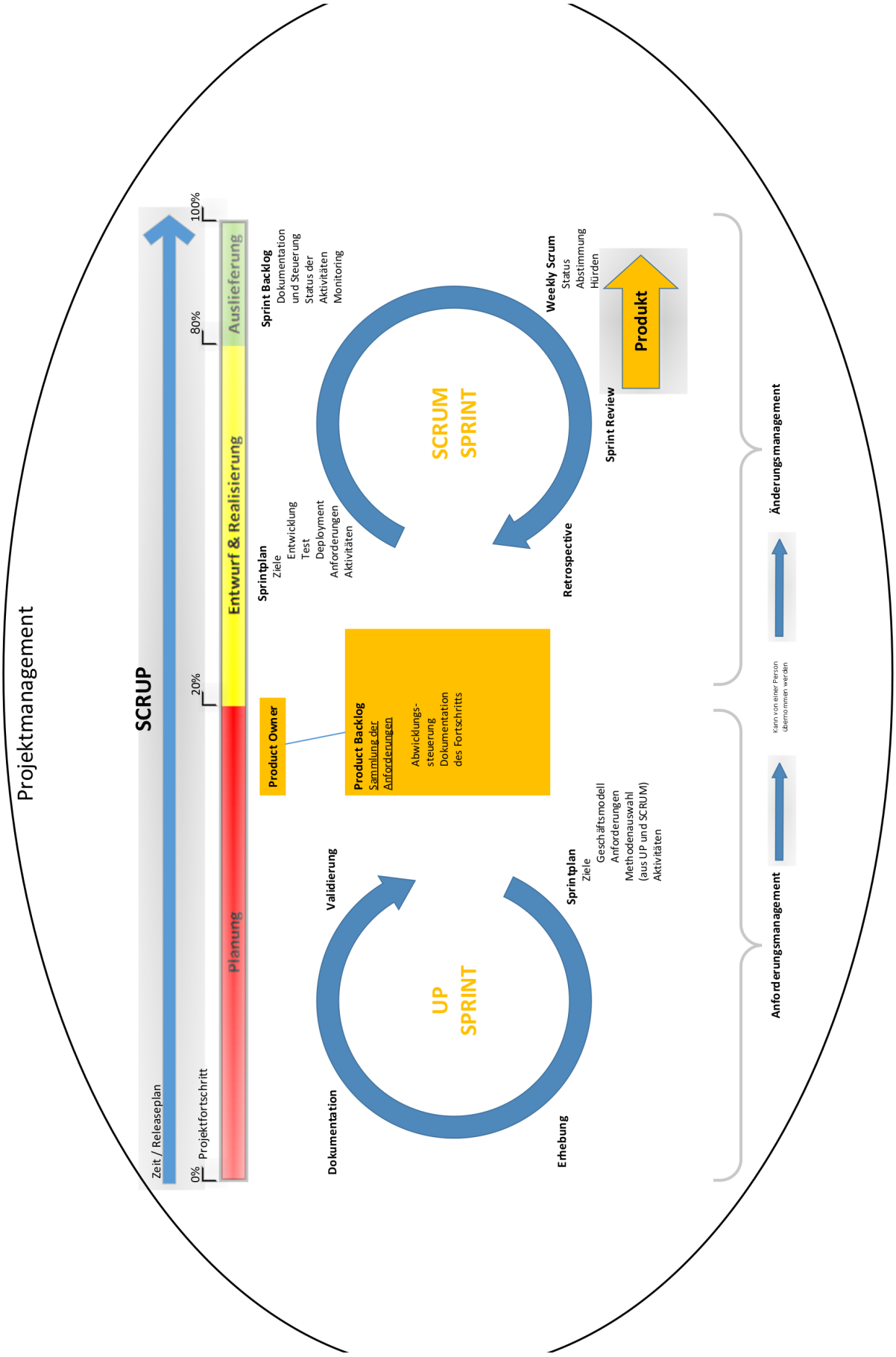


Abbildung 1: Übersicht des Vorgehensmodells ScrUP

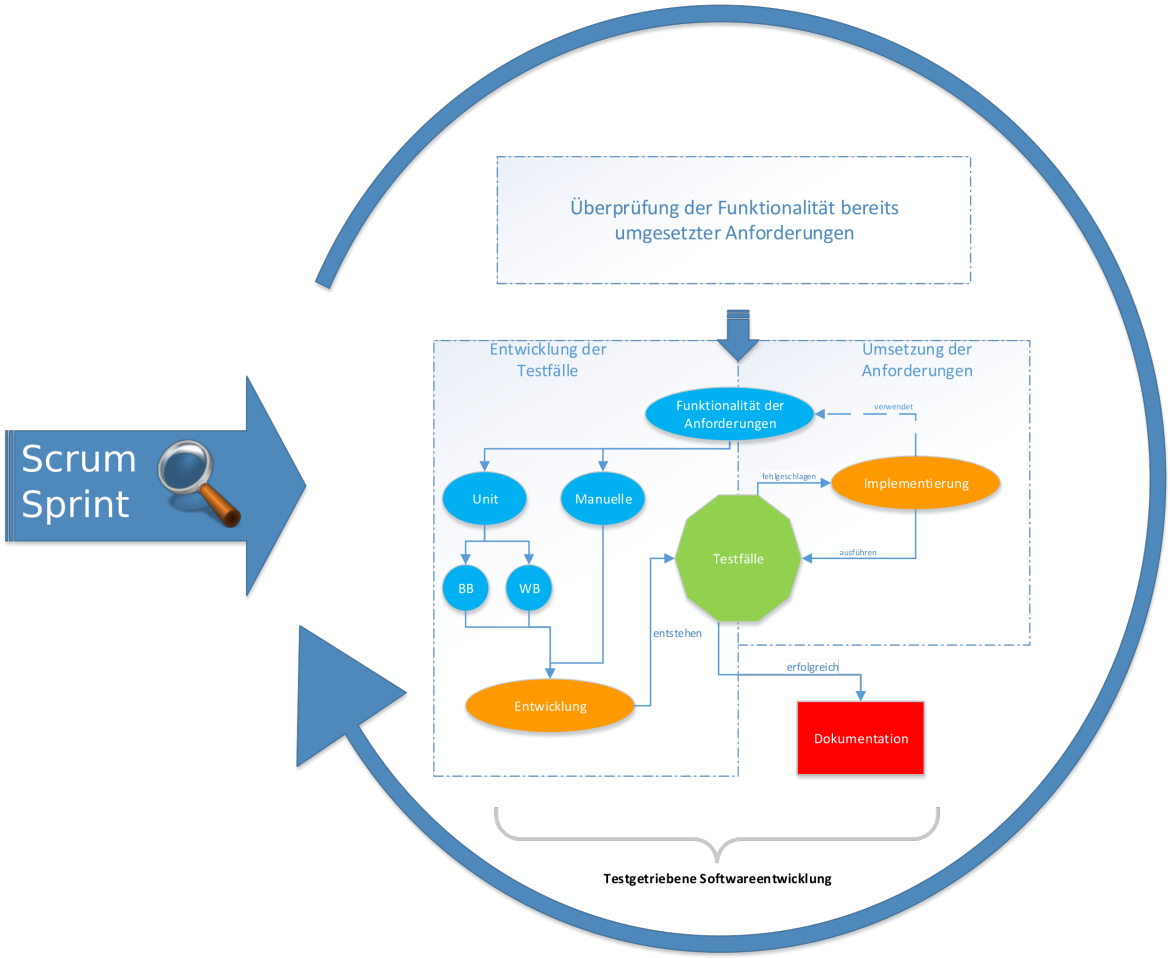


Abbildung 2: Details eines Scrum Sprints mit testgetriebener Softwareentwicklung im Vorgehensmodell ScrUP

Vorangestellte Planungsphase Im Gegensatz zum üblichen Scrum-Modell wird in diesem Vorgehensmodell eine Planungsphase vorangestellt. Diese ist mit der *Inception*- und *Elaboration*-Phase des *Unified Process* vergleichbar. Hier wird mit den bekannten *Unified Process*-Techniken vorgegangen, um eine Anforderungserhebung durchzuführen. Dieser erste Teil dient der Vorbereitung der Entwurfsphase, welche im Anschluss direkt beginnen kann.

Sprint Eine weitere Unterscheidung in der Durchführung besteht während des Sprints selbst. Durch die zeitliche Beschänkung kann nicht jedes Projektgruppenmitglied täglich mitarbeiten kann. Daher wird es als unrealistisch und nicht notwendig eingeschätzt sich täglich zu treffen. Es wird somit auf ein *Daily Scrum* verzichtet. Als Ausgleich dazu werden zwei Treffen pro Woche abgehalten, während derer über Fortschritt und Hürden berichtet wird. Unter anderem wird auch während dieser Treffen der jeweilige Sprint geplant und ein formloser Bericht aller Mitglieder über ihre jeweiligen erledigten und anstehenden Aktivitäten durchgeführt. Die Länge eines Sprints wird zwischen einer und drei Wochen festgelegt. Die Menge der aufzunehmenden Aufgaben ist von der Sprintlänge abhängig.

Testgetriebene Entwicklung Auf Grund der Verschmelzung von Scrum und *Unified Process* ist eine testgetriebene Entwicklung unabdingbar. Das genaue Vorgehen wird im rechten Teil von Abbildung 1 durch ein Flussdiagramm dargestellt. Es zeigt auf, dass zunächst aus den schon bestehenden Anforderungen Testfälle entstehen, welche implementiert und dokumentiert werden. Die weitere Umsetzung der testgetriebenen Entwicklung wird später in Kapitel 7 erläutert.

Verkürzte Auslieferung und Betreuung Durch den festen Zeitrahmen des Studierendenprojektes ist keine ausgedehnte Auslieferung durchzuführen. Zwar kann dem Auftraggeber das Projekt übergeben und funktional aufgesetzt werden, allerdings ist es nicht möglich, dieses über den angesetzten Zeitraum des Projektes hinaus zu betreuen.

Erweiterte Retrospektive Im Verlauf des Projektes wurde nicht nur eine Retrospektive über die Projektarbeit durchgeführt, sondern auch das eigene Vorgehen stetig evaluiert, um die Qualität der Arbeit aufrecht zu erhalten und zu verbessern. Details dazu finden sich im Projektablauf. Die aus dem Scrum bekannten Rollen des *Scrum Masters* und *Product Owners*, die in Kapitel 2.3.1 näher beschrieben werden, wurden teilweise auf mehrere Personen verteilt. Dies sollte zu einer Entlastung der einzelnen Personen führen. Zusätzlich wurden die Rollen nach der Hälfte des Zeitraums neu besetzt. So wurde mehr Teilnehmern ermöglicht, Erfahrungen in diesen Rollen zu erhalten.

Change Management Während im Unified Process das Anforderungsmanagement eine übergeordnete Aufgabe zur Steuerung und Kontrolle des Anforderungserhebungsprozesses ist, übernimmt das Änderungsmanagement eine sehr ähnliche Rolle im Scrum. In beiden Fällen muss ein Überblick über den aktuellen Stand geschaffen und zudem der Zeitrahmen, sowie die weiteren zur Verfügung stehenden Ressourcen organisiert werden. Das Scrup-Modell kombiniert Eigenschaften von Scrum und Unified Process und dieses Management muss in diesem Scrup-Modell die Sprints überwachen und steuern. Zudem muss es für bestimmte Zeitpunkte festlegen, ob Projektabschnitte abgeschlossen sind und in den nächsten Schritt übergegangen wird.

2.2. Projektablauf

Dieses Kapitel beschreibt im Überblick den Ablauf des Projekts von Beginn bis Ende und betrachtet dabei die Aspekte, die zur Planung und Steuerung des Projekts umgesetzt wurden.

Der zeitliche Rahmen der Projektgruppe erstreckte sich vom 1. April 2015 bis zum 31. März 2016. Das erste Treffen der gesamten Projektgruppe fand am 9. April bei der Abschlusspräsentation der vorherigen Projektgruppe statt. Im Anschluss wurde eine grobe thematische Vorstellung des neuen Projektgruppenthemas gegeben und erste organisatorische Angelegenheiten geklärt. So wurden beispielsweise die Themen für die dem eigentlichen Projektstart vorangehende Seminarphase verteilt, welche innerhalb der ersten drei Wochen in Einzelarbeit behandelt wurden und anschließend an zwei Vortragstagen (8. und 9. Mai 2015) vorgestellt wurden. Die Seminarphase zu Beginn stellte einen schnellen Einstieg in die Thematik der Projektgruppe sicher. Zudem ergab die Seminarphase eine erste Spezialisierung der einzelnen Projektmitglieder. Die Ergebnisse sind im Anhang G als ein Seminarband angefügt.

Das erste reguläre Treffen fand am 13. Mai 2015 statt. Bei diesem wurde festgelegt, dass im weiteren Ablauf zunächst zwei Treffen (die im Folgenden auch als Sitzungen oder *Meetings* bezeichnet werden) pro Woche stattfinden sollen. Bei einem dieser Meetings sind die Projektbetreuer zeitweise ebenfalls anwesend. Während der Pojektlaufzeit änderten sich die freien Zeitfenster der Projektmitglieder abhängig von Klausurephase, Vorlesungszeit und vorlesungsfreier Zeit. In der Zeit der Klausurenphasen konnte weniger Zeit für das Projekt aufgewendet werden. Stattdessen konzentrierte sich in den vorlesungsfreien Zeiten der Großteil der Arbeitskraft auf das Projekt. Dementsprechend änderten sich auch die Sitzungstermine, es blieb zunächst jedoch bei zwei Meetings pro Woche. Zum Ende der Projektlaufzeit änderte sich auch die Sitzungshäufigkeit auf drei Meetings pro Woche, da in der Abschlussphase deutlich mehr Arbeitskraft in das Projekt investiert werden sollte. Dabei wurde die Häufigkeit der Sitzungen mit den Projektbetreuern jedoch nicht geändert. Außerdem wurde anfangs ein Vorgehensmodell entwickelt, welches zum einen auf das in einem Seminarbeitrag unter Kapitel 11 in Anhang G vorgestellte *Scrum* und zum anderen auf *Unified Process* aufbaut. Das genaue Ergebnis dieser Kombination wird im Abschnitt über das Vorgehensmodell vorgestellt.

Die wöchentlichen Sitzungen wurden jeweils durch eine Tagesordnung strukturiert. Im Verlauf des Projektes haben sich einige statische Tagesordnungspunkte ergeben, wobei die Tagesordnungspunkte 4 bis 6 nur am Ende eines Sprints aufgegriffen wurden.

1. Organisatorisches
   - Begrüßung
   - Ergänzungen der Tagesordnung
   - Genehmigung des letzten Protokolls
2. Standup 1
   - Jede(r) sagt was er/sie gemacht hat
3. Berichte und Diskussionen
   - Generell: nur kurze Berichte mit kurzer Diskussion
     - falls Diskusionen zu lange dauern: in Kleingruppen auslagern
4. Sprint Review
   - Jira Tasks ins Protokoll übernehmen
5. Sprint Retrospektive
   - in positive und negative Aspekte unterteilen
   - negativen Aspekten im Anschluss / direkt Schlussfolgerungen in einem Unterpunkt zuordnen
   - Der Protokollant übernimmt diese ins Confluence (nach der Sitzung)!
6. Sprintplanung
   - (Planning-Poker)

- Auswahl zu realisierender Backlog Items
- Tasks in Jira erstellen
- Aufgabenverteilung

7. Sonstiges
   - Urlaubsankündigungen
8. TODO
   - Tasks die während der Sitzung anfallen werden hier notiert und im entsprechenden Segment vergeben / in Tasks umgewandelt / in den neuen Sprint aufgenommen
9. Standup 2
   - Jede(r) sagt, was sie/er als nächstes tut
10. Retrospektive der Sitzung
    - positiv
    - negativ
    - Schlussfolgerung

Die Sitzung wird von einem Moderator geleitet und von zwei Protokollanten verschriftlicht. Diese Rollen werden im Kapitel Rollen näher beschrieben.

Die Schlussfolgerungen aus den Retrospektiven haben zu vielen konstruktiven Verbesserungen geführt. Viele haben zu Richtlinien oder Empfehlungen geführt, die während der Sitzung zunächst im Etherpad (s. Abschnitt 2.5.3) und anschließend dauerhaft in Confluence (s. Abschnitt 2.5.4) festgehalten wurden. Für eine bessere Zeitplanung der Sitzung wurde die Zeitschätzung für jeden Tagesordnungspunkt eingeführt. Außerdem konnten längere Diskussionen vom Moderator in Kleingruppen ausgelagert werden, um die Arbeitseffizienz zu steigern.

Neben den regulären Sitzungen wurden für umfangreichere Aufgaben auch Workshops durchgeführt. Dazu zählen z. B. der Anforderungsworkshop (s. Abschnitt 4.1) und gemeinsame Codereviews (s. Abschnitt 2.4).

Für das Zeitmanagement der Projektmitglieder war vorgesehen, dass jeder bis zu sechs Wochen Urlaubszeit beanspruchen durfte, wobei zwei dieser Wochen für die Weihnachtsferien genutzt werden mussten. Die Urlaubszeiten wurden in einem Kalender eingetragen und zusätzlich in der vorhergehenden Sitzung angekündigt. Auf Grundlage der 24 Kreditpunkte ergaben sich 16 Stunden pro Woche und Projektmitglied als empfohlene Richtarbeitsdauer. Mit der Einführung der Kernzeiten zum Ende der Projektlaufzeit wurde eine Mindestarbeitszeit von 21 Stunden pro Woche je Projektmitglied sowie Kernzeiten von 9 bis 15 Uhr jeden Dienstag, Mittwoch und Donnerstag für Februar und März 2016 festgelegt. In den Kernzeiten hatte jedes Mitglied, bis auf Ausnahmen wie Urlaub und andere triftige Gründe, anwesend zu sein.

Begleitend wurden in Kleingruppen mehrfach Netzpläne mit den zu erledigenden Vorgängen bzw. Tasks und Meilensteinen erstellt und gepflegt, die einen Überblick über den Aufwand und die Termin-

fristen gaben. Diese lassen sich in Anhang [C] einsehen. Details zu den Inhalten der Tasks und Sprints finden sich in den monatlichen Sprintzusammenfassungen in Anhang [B] und im JIRA-Projekt.

Neben den bisher genannten waren weitere Planungsaktivitäten aufgrund der sehr strukturierten und disziplinierten Arbeitsweise und der stetigen Reflexionen und Retrospektiven der Sprints in der Projektgruppe nicht notwendig. Dazu zählt beispielsweise das Erstellen von Projektstrukturplänen oder eine detailliertere Zeit- und Ressourcenplanung als die, die innerhalb der Sitzungen und Kleingruppen stattfand.

Während des Projekts hat zunächst das Anforderungsmanagement und später das Änderungsmanagement dafür gesorgt, dass Projektphasen kontrolliert ablaufen und abgeschlossen wurden. Besonders in der Schlussphase, in der es um den Rest der Implementierung, die Fehlerbehebung und die Dokumentation ging, kamen einige Änderungsanfragen auf. Um gesetzte Projektziele zu erfüllen und geplante Funktionen umzusetzen, wurde deren Umsetzung als noch notwendig erachtet. Zudem kamen Änderungen aufgrund von unterschiedlichem Verständnis von Funktionsdetails zwischen Projektgruppe und Auftraggebern auf. Dazu gehören Details der Optimierung, der Zusammensetzung von Einsatzplänen und Produktportfolios sowie der Preis- und Kostenberechnung. In der Schlussphase wurden nur noch kleinere Änderungen am Projekt durchgeführt, die für die Funktionalität von großer Bedeutung waren. Die Relevanz der Änderungen wurde bei den Schätzungen als *nice-to-have* oder *must-have* eingeschätzt.

Sprints

Das ScrUP-Vorgehensmodell sieht die Benutzung der aus Scrum bekannten Sprints vor. Dies ist ein Arbeitsschritt, in dem inkrementell das Produkt verbessert wird. Er beginnt mit dem *Sprint Planning* und endet mit einem *Sprint Review*, bei welchem die geschafften und nicht geschafften Aufgaben zusammengefasst werden und besteht weiterhin aus der *Sprint Retrospektive*, bei der auf kommunikativer, organisatorischer und struktureller Ebene auf den Sprint zurückgeblickt wird. Im Projekt wurde zunächst eine zweiwöchige Sprintlänge festgelegt. Zusätzlich zur Anpassung der Sitzungstermine über den Projektzeitraum wurde auch die Länge der Sprints aus dem entwickelten Vorgehensmodell einmalig geändert. Bis etwa Anfang September 2015 waren die Sprints zweiwöchig, dann wurde mit Beginn der Implementierung auf einwöchige Sprints gewechselt, da der Eindruck entstand, dass Aufgaben bevorzugt am Ende des Sprints erledigt werden. Zudem versprachen kürzere Sprints mehr Überblick und feingranularere Aufgaben, da, wie zuvor erwähnt, die Mitglieder zum Ende hin wesentlich mehr Zeit für das Projekt aufwenden konnten. Mit den festen Kernzeiten zum Ende der Projektlaufzeit wurde zwar die Sitzungshäufigkeit auf drei Meetings pro Woche geändert, die Sprintlänge von einer Woche wurde jedoch beibehalten und somit waren auch nur jeweils ein *Sprint Planning*, ein *Sprint Review* und eine *Sprint Retrospektive* pro Woche nötig.

Entstehung des Projektakronyms *Powder*

Das Akronym *Powder* für das Projekt entstand während eines der vielen Meetings aus der Funktion des zu entwickelnden Systems heraus. *Powder* steht für *Profit Optimization with Distributed Energy*

Resources. Das ausformulierte Akronym beschreibt damit selbst, was das System tut und ist gleichzeitig der selbst ernannte Name der Projektgruppe.

2.3. Rollen

Dieser Abschnitt beschreibt die eingeführten Rollen, die von der Projektgruppe als wichtig erachtet wurden. Die Rollen sind dabei nicht nur von einer Person ausgeführt worden, sondern wurden zur Hälfte der Projektzeit neu besetzt. Dies gewährleistet, dass jedes Projektmitglied einen Gesamtüberblick über die Aufgaben bekommt und die Last von arbeitsaufwändigeren Rollen aufgeteilt wird. Da sich die Projektgruppe auf eine Vielzahl von Rollen geeinigt hat, wurden diese aus Gründen der Übersicht in der Dokumentation in drei Kategorien aufgeteilt.

Die erste Kategorie beinhaltet dabei alle Rollen, die sich direkt mit der Projektplanung und dem in Abschnitt 2.1 beschriebenen ScrUP-Vorgehen befassen, wie beispielsweise Sitzungsleiter, Protokollanten oder Leiter der Anforderungserhebung. In der zweiten Kategorie werden alle Rollen zusammengefasst, die sich mit einem speziellen Bereich oder einer speziellen Aufgabe befassen. Diese werden als Bereichsrollen für die Dokumentation definiert. Darunter fallen beispielsweise ein Beauftragter für das verwendete Textverarbeitungsprogramm, sowie für weitere verwendete Programme (Git, JIRA, usw.). In der dritten Kategorie werden alle weiteren Rollen zusammengefasst. Darunter werden Rollen verstanden, die nur indirekt mit dem eigentlichen Projekt in Verbindung stehen, jedoch von der Projektgruppe als wichtig erachtet wurden. Dazu gehören beispielsweise die Schlüsselträger und ein Beauftragter für Gruppenaktivitäten. Für alle Rollen wurde festgelegt, dass die Information, welcher Projektteilnehmer zu welchem Zeitpunkt welche Rolle innehatte, nicht benannt wird. Diese Entscheidung wurde von der Projektgruppe intern beschlossen. Als Begründung wurde angegeben, dass nicht jeder eine Rolle besetzen kann, da es nicht genügend Rollen gibt und Projektmitglieder ohne Rolle dadurch nicht schlechter gestellt werden sollen. Weiterhin wurde von einigen Projektmitgliedern der Punkt des individuellen Datenschutzes aufgeführt.

2.3.1. Rollen des ScrUP-Prozesses und der Projektplanung

Bei den Rollen der Projektplanung und des ScrUP-Prozesses handelt es sich wie erläutert um die Rollen, die für diesen Bereich benötigt werden. Dabei kann es sich bei der Projektplanung jedoch um Rollen handeln, die nur für den jeweiligen Projektabschnitt von Bedeutung sind. Im Folgenden werden alle Rollen dieser Kategorie genannt und kurz beschrieben.

Die erste Rolle im Projektablauf ist der *Moderator*. Er hat die Aufgabe, den Vorsitz einer Projektgruppensitzung zu übernehmen, Abstimmungsergebnisse zu forcieren und Diskussionen zu leiten. Zudem müssen organisatorische Aufgaben wie die Erstellung der Einladung zur Sitzung sowie der Tagesordnung durch ihn ausgeführt werden. Die Rolle des Moderators wird jedoch nicht, wie viele andere, für einen bestimmten Zeitraum festgelegt, sondern zu jeder Sitzung gewechselt. Dabei wird eine festgelegte Moderationsreihenfolge eingehalten. Die Entscheidung, dass jedes Projektmitglied die Rolle des Moderators ausgeführt haben soll, kann damit begründet werden, dass die Fähigkeit der Gruppenleitung erprobt und gefördert werden soll. Außerdem ist dadurch gewährleistet, dass jeder

die Chance bekommt, seine eigenen Ideen und Vorgehensweisen während einer Sitzung integrieren zu können.

Der *Protokollant* ist die zweite wichtige Rolle im Bereich der Projektsitzungen. Die Aufgaben beziehen sich dabei auf das Festhalten der wichtigsten Ereignisse in einer Sitzung sowie der Dokumentation der beschlossenen Ergebnisse. Als organisatorische Aufgaben fallen zudem die Überarbeitung und das Versenden des Sitzungsprotokolls in den Aufgabenbereich des Protokollanten. Der Protokollant wird bei seiner Arbeit während der Sitzung von einem Zweitprotokollanten unterstützt. Auch diese Rolle wechselt zu jeder Sitzung und hängt dabei auch von der Moderationsreihenfolge ab, da jeder Moderator in der vorherigen Sitzung als Protokollant und davor als Zweitprotokollant eingeteilt wird.

Neben den Rollen für die wöchentlichen Sitzungen werden auch die typischen Rollen des Scrum-Prozesses von der Projektgruppe verwendet. Der *Product Owner* ist eine dieser Rollen. Dieser muss den *Product Backlog*, welcher die Aufgaben für den nächsten Sprint enthält, mit Aufgaben füllen und diese Aufgaben auch priorisieren. Außerdem steht diese Rolle in Verbindung mit den Stakeholdern und vertritt deren fachliche Auftraggeberseite. Da diese Aufgaben eine komplette Übersicht über das Vorgehen des Projekts benötigen, wurde aus Komplexitätsgründen die Rolle auf zwei Projektmitglieder aufgeteilt.

Die zweite Rolle des Scrum-Prozesses ist der *Scrum Master*. Dieser dient als Vermittler und Unterstützer der Projektgruppe während eines Scrum-Sprints, sowie als Aufsichtsperson über den Scrum-Prozess und den *Product Backlog*. Weiterhin moderiert dieser die Sprint-Meetings. Da jedoch die Sprint-Meetings mit den wöchentlichen Sitzungen äquivalent zu setzen sind, wird die Aufgabe der Sitzungsleitung vom selben Moderator übernommen. Für die Projektgruppe besitzt der *Scrum Master* in dem jeweiligen ScrUP-Prozess die Aufgabe der Überwachung des Prozesses. Außerdem bereinigt er Unklarheiten über einen ScrUP-Prozess. Die Rolle wurde zum Zeitpunkt der Abgabe des Zwischenberichts durch ein anderes Projektmitglied besetzt, welches sich aber weiterhin mit dem vorherigen *Scrum Master* abgesprochen hat.

Die erste Phase des Projektablaufs wurde durch die Projektgruppe als *Requirements Engineering* benannt. Für diese Phase wurde die gleichnamige Rolle *Requirements-Engineering-Beauftragter* entwickelt, die zur Aufgabe hat, den Ablauf des *Requirements Engineering* zu Planen und der Projektgruppe als Berater in Fragen zum Ablauf zur Seite zu stehen. Bei dieser Rolle handelt es sich um eine zeitbegrenzte Rolle, da diese nur solange benötigt wird, bis diese Phase abgeschlossen wurde. Anschließend wird diese Rolle durch das Change-Management ersetzt, wie dem Vorgehensmodell zu entnehmen ist. Da auch diese Rolle ein komplexes Aufgabenfeld besitzt, wurde auch diese auf zwei Mitglieder aufgeteilt.

Als die letzte Rolle in diesem Bereich wurden die *Lektoren* festgelegt. Diese haben zur Aufgabe, wichtige Dokumente auf den Lesefluss, die inhaltliche und formale Konsistenz, die Einhaltung der definierten Schreibkonventionen sowie syntaktische Fehler zu überprüfen und ggf. zu korrigieren oder als neue Aufgabe zu delegieren. Dieses Vorgehen dient zudem der Qualitätssicherung (s. Abschnitt 2.4).

Diese Rolle wurde aufgenommen, da es durch die zahlreichen und unterschiedlichen Autoren erfahrungsgemäß zu einem schwierigeren Lesefluss und vielen syntaktischen und semantischen Fehlern

kommt. Dies soll durch die Lektoren verhindert werden. Diese Rolle wird aufgrund ihres Umfangs von einer Gruppe übernommen und wird nur bei der Erstellung von wichtigen Dokumenten benötigt.

2.3.2. Bereichsrollen

Neben den Rollen, die den Ablauf des Projektes gewährleisten sollen, wurden von der Projektgruppe weitere Rollen beschlossen, die sich mit bestimmten Bereichen oder auch Software befassen.

In der Projektgruppe wurde für den Projektablauf die Verwendung des Softwareentwicklungstools *JIRA* (s. Abschnitt 2.5.2) vereinbart. Dafür wurde zudem eine Rolle erstellt, der *JIRA-Beauftragte*. Dieser besitzt Administratorrechte in JIRA und sorgt dafür, dass die Projektmitglieder die Aufgaben bzw. *JIRA-Tasks* richtig erstellen. Zudem ist er für das Verteilen von Rechten zuständig. Weiterhin dient der JIRA-Beauftragte auch als Berater bei Problemen mit JIRA. Diese Rolle wird aufgrund der Projektverantwortung und des Aufwands von zwei Projektmitgliedern ausgeführt.

Da in den Sitzungen der Projektgruppe beschlossen wurde, testgetriebenes Programmieren durchzuführen, wurde die Rolle des *Testbeauftragten* eingeführt. Dieser hat zur Hauptaufgabe, die Evaluation der möglichen Softwaretools zur testgetriebenen Programmierung durchzuführen, sowie zu kontrollieren, ob die Tests geschrieben und durchgeführt werden. Die Rolle wird ebenfalls auf Grund des Aufwands von zwei Personen ausgeführt.

Da die meisten verwendeten Anwendungen, wie Bamboo oder JIRA, zunächst einmal aufgesetzt werden mussten oder für diese bestimmte Plug-ins benötigt wurden, wurde die Rolle des *Integrationsbeauftragten* beschlossen. Dieser hat die Aufgabe, die beschlossenen Anwendungen und Plug-ins zu integrieren oder aufzusetzen. Zudem verteilt dieser auch die Aufgaben, *Kurzanleitungen* dazu zu schreiben.

Die Projektgruppe arbeitet überwiegend mit zwei Git-Repositorien. Darüber hinaus wird zum dauerhaften Speichern der Sitzungsprotokolle ein Confluence-Repositorium verwendet. Die Rolle des *Repository-Beauftragten* soll das Einpflegen der Texte überwachen und den Projektmitgliedern bei Problemen im Bereich der Repositorien helfen.

Da die meisten Dokumente in der Projektgruppe mit LaTeX erstellt werden, wurde ein *LaTeX-Beauftragter* gewählt. Dieser hat zur Aufgabe, ein LaTeX-Meta-Dokument zu erstellen, in dem alle Konventionen für das Schreiben von LaTeX-Dokumenten beinhaltet sind. Des Weiteren legt dieser Dokumente an und dient als Ansprechpartner bei Problemen mit LaTeX.

2.3.3. Sonstige Rollen

Als sonstige Rollen wurden solche definiert, die nicht direkt mit dem Projektablauf in Beziehung stehen.

Die Aufgabe der *Schlüsselträger* verrät bereits der Name. Da es nur eine bestimmte Anzahl von Schlüsseln für den Projektraum gibt, wurden diese unter den Projektmitgliedern aufgeteilt.

Da die Projektgruppe für einen längeren Zeitraum zusammenarbeiten soll, wurde die Rolle des *Sozialplanbeauftragten* erfunden. In dessen Zuständigkeitsbereich liegt die Koordination und Organisation von gemeinsamen Aktivitäten außerhalb des Projektes.

Die letzte Rolle ist der *Kaffeebeauftragte*. Dieser organisiert warme und kalte Getränke, die gerne von der Projektgruppe konsumiert werden und sammelt die entsprechenden Kosten der Beschaffung dafür ein.

Neben den verwendeten Rollen wurden zudem in der Endphase der Projektgruppe Modulverantwortliche bestimmt, die im Folgenden betrachtet werden.

2.3.4. Modulverantwortliche

Die Modulverantwortlichen wurden aus Gründen der Einhaltung des Zeitplans sowie Umsetzung der Fertigstellung der einzelnen Module (s. Abschnitt 6.1) bzw. der wichtigen übergeordneten Aufgaben des Projekts eingeführt. Damit trotz der hohen Komplexität genügend Übersicht über das gesamte Projekt vorhanden ist, wurden die Modulverantwortlichen damit beauftragt, in ihren Bereichen alle noch auszuführenden Aufgaben zu ermitteln und ggf. an andere Mitglieder zu delegieren. Weiterhin sollen die Modulverantwortlichen darauf achten, dass Zeitpläne eingehalten werden. Auch besitzen die Modulverantwortlichen die Kontrolle über die Dokumentation des ihnen verantwortlichen Moduls. In der folgenden Aufzählung sind die Verantwortungsbereiche der Modulverantwortlichen aufgelistet:

- VPPMain
- Flexibilitäten
- Wetter
- Markt
- Evaluation
- Optimierer / PPO
- Optimierer / EPO
- OPC UA/ externe Kommunikation
- DB Connection / lokale DB)
- Shared / RMI / interne Kommunikation
- GUI
- Dokuverantwortlicher

Da genau zwölf Bereiche vorhanden sind, wurde jedem Projektmitglied ein Bereich zugewiesen.

2.4. Qualitätsmanagement

Das Qualitätsmanagement gehört zu einem der zentralen Bereiche im Projektmanagement. Für das Projekt kann das Qualitätsmanagement in unterschiedliche Bereiche eingeteilt werden. Dabei handelt es sich um ein kontinuierliches Qualitätsmanagement, sowie ein Qualitätsmanagement für jeweilige Projektabschnitte bzw. für bestimmte Abgabeartefakte.

Da es sich eher um ein kleineres Projekt handelt und die meisten Qualitätsmanagement-Konzepte sehr komplex sind, wurde sich in einer Abstimmung in der Projektgruppe auf ein auf die eigenen Bedürfnisse angepasstes Qualitätsmanagement-Konzept festgelegt, so dass für jeden Bereich speziell betrachtet wurde, welche Werkzeuge in welchem Umfang eingesetzt werden sollten. Es werden also keine Standardverfahren des Qualitätsmanagements wie ISO 10006, welches den meisten Forderungen der ISO 9001 entspricht, oder Capability Maturity Model Integration (CMMI) verwendet, sondern es wird nur auf einige Punkte und Werkzeuge dieser Verfahren, die besonders wichtig erscheinen, eingegangen.

Im Bereich des *kontinuierlichen Qualitätsmanagements* liegt die Aufgabe vor allem auf der Erhaltung und Verbesserung der Sitzungsqualität. Dafür wurde nach einigen Sitzungen beschlossen, am Ende jeder Sitzung eine *Retrospektive* für den Sprint und die Sitzung durchzuführen. Dabei werden für jede Sitzung positive und negative Punkte gesammelt, die die Durchführung der jeweiligen Sitzung bewerten (*Sitzungsretrospektive*) . Für die negativen Punkte werden daraufhin Schlussfolgerungen gezogen, damit der gleiche negative Punkt nicht wieder auftaucht. So soll gewährleistet werden, dass sowohl die Sitzungen verbessert werden, als auch die einzelnen Moderatoren Verbesserungen an ihrem Moderationsstil vornehmen können. Bei der Retrospektive für den Sprint (Sprintretrospektive) wird der Ablauf des Sprints betrachtet. Es werden positive und hinderliche Faktoren festgehalten und Lösungen für aufgetretene Probleme gesucht.

Neben der Qualität der Sitzungen zählt zu den Aufgaben im kontinuierlichen Qualitätsmanagement, die Code-Qualität zu erhalten und zu verbessern. Für das Code-Qualitätsmanagement wurde beschlossen, in gewissen Abständen *Codereviews* durchzuführen, in denen die Hauptmodule im Code durchgegangen werden. In diesem Zusammenhang wird dabei vor allem über Verbesserungen und Konventionen im Programmierstil gesprochen und diese dann auch vereinbart. Aus diesen Vereinbarungen werden dann neue Aufgaben zur Verbesserung der relevanten Code-Stellen erstellt. Des Weiteren wurde durch die Projektgruppe beschlossen, eine interne *Code-Präsentation* durchzuführen. Dabei sollten die einzelnen Module durch Projektmitglieder präsentiert werden, die sich nicht direkt damit beschäftigt haben. Dadurch soll gewährleistet werden, dass sich jeder mit dem Großteil des Codes befasst und Unklarheiten im Code bzw. der Codedokumentation besprochen und bei Bedarf angepasst werden können. Durch diese Präsentation soll, wie im Codereview, die Qualität des Codes betrachtet werden. Bei allen diesen Reviews und Präsentationen wurden auch die Projektbetreuer eingeladen, damit auch diese ihre Ideen und Anmerkungen an den Code abgeben konnten. Außerdem wurde von der Projektgruppe beschlossen, eine testgetriebene Implementierung durchzuführen (s. Abschnitt 7), durch die auch gewährleistet werden soll, dass der Code auch die richtigen Ergebnisse und Befehle ausgibt.

Das erste separate und nicht kontinuierliche Qualitätsmanagement-Konzept wurde für die *Requirements Engineering*-Phase entwickelt. Während der Anforderungserhebung (s. Abschnitt 4) nach dem

SGAM-Vorgehensmodell (s. Abschnitt 4.1 und Seminarband), wurden nach Erstellung von Artefakten *Konsistenzgruppen* gebildet. Diese Gruppen mussten das SGAM mit der Anforderungsdokumentation in Konsistenz bringen, wodurch auch die Qualität der SGAM-Dokumentation positiv beeinflusst und gesichert wurde. Des Weiteren wurde durch die kontinuierliche Vorstellung der SGAM-Ergebnisse für die Betreuer eine Grundlage geschaffen, in der über die Richtigkeit des SGAM gesprochen werden und erkannte Fehler behoben werden konnten.

Für die Phase der Implementierung und der Abgabe des Produkts wurde neben dem Einsatz der bereits beschriebenen Codereviews auch die Codedokumentation überprüft. Dafür wurde ein Projektmitglied abgestellt, welches die Javadoc überprüfen und ggf. Verbesserungen delegieren sollte. Zudem wurden für die letzte Phase des Projekts (die letzten 3 Monate) Modulverantwortliche (s. Abschnitt 2.3.4) bestimmt, welchen jeweils ein Verantwortungsbereich auf Basis der Softwaremodule zugewiesen wurde. Innerhalb dieses Verantwortungsbereichs steuerte jeder Verantwortliche den Fortschritt. Darunter fiel sowohl die Sicherung der Qualität des Moduls im Code, aber auch auf Seiten der Dokumentation.

Für die Abgabe von größeren Dokumenten wurden in der Projektgruppe *Lektoren* eingeführt (s. Abschnitt 2.3). Für die Erstellung von wichtigen Präsentationen wurde zudem vereinbart, mindestens einen internen Vorvortrag abzuhalten und in einem Review zu besprechen und anzupassen.

Unter den letzten Qualitätsmanagementbereich fallen alle *sonstigen Qualitätsmanagementaufgaben*. Dazu gehört eine Evaluation der Arbeitsvoraussetzungen bzw. der Infrastruktur für die Arbeit der Projektmitglieder. Dies kann als Qualitätsprüfung aufgefasst werden. Dazu wurde von der Projektgruppe eine Evaluation der eingesetzten Technologien beschlossen. Das Ergebnis dieser Evaluation ist in der Evaluation der Technologien (s. Abschnitt 5.15) zu finden. Diese Evaluation soll vor allem der Erkennung von nicht genutztem Potenzial der Technologien sowie der Behebung einzelner individueller Probleme dienen. Da die Projektgruppe nur für ein Jahr besteht, dient diese Evaluation jedoch nicht als Anregung, bei schlechten Ergebnissen für eine Technologie auf eine äquivalente Technologie umzusteigen.

Die verschiedenen Bereiche zeigen, dass von der Projektgruppe ein auf die Bedürfnisse abgestimmtes Qualitätsmanagement durchgeführt wurde. Mit mehr Vorwissen und bei einem größeren Projekt hätte das Qualitätsmanagement auch noch wesentlich stärker ausfallen können, trotzdem wurden die wichtigsten Bereiche durch das spezielle Qualitätsmanagement durch die Projektgruppe abgedeckt und auch durchgeführt.

2.5. Infrastruktur

Ebenso wichtig wie Vorgehensmodell, Rollenverteilung und Qualitätsmanagement ist der Aufbau der Projektinfrastruktur. Insbesondere von Bedeutung sind eine Versionsverwaltung für Dokumente sowie Kommunikationsmöglichkeiten auch außerhalb von offiziellen Treffen. Zusätzlich kann eine Projektmanagement-Software hilfreich sein. Im Folgenden werden verschiedene Werkzeuge vorgestellt, die von der Projektgruppe im alltäglichen Gebrauch und zur Organisation des Projektablaufs verwendet wurden.

2.5.1. Git

Bei der Wahl einer Software zur Versionsverwaltung fiel die Entscheidung auf Git. Zum einen, weil einige Teammitglieder gute Erfahrungen mit Git sowie schlechte Erfahrungen mit SVN gesammelt haben. Zum anderen, weil Git eine der modernsten und populärsten Softwares zur Versionsverwaltung ist. Im Gegensatz zu SVN verfolgt Git einen dezentralen Ansatz, woraus sich auch die Möglichkeit zum Arbeiten ohne Internetverbindung ergibt.

Bei Git werden Dateien in einem Repositorium verwaltet und dessen Versionsgeschichte in einem Git-Log gespeichert. Dies ermöglicht z. B. das Zurückzusetzen von Dateien auf einen alten Stand. In der Regel ist es kein Problem, wenn mehrere Personen am gleichen Projekt oder sogar an der gleichen Datei arbeiten. Ein Problem ergibt sich erst, wenn in der gleichen Zeile der gleichen Datei gearbeitet wird. In diesem Fall kommt es beim Zusammenführen der Versionen zu Konflikten, die manuell gelöst werden müssen.

Für Code und Dokumente wurden jeweils ein eigenes Ropositorium angelegt, damit die Git-Logs übersichtlicher sind.

2.5.2. JIRA

Die webbasierte Anwendung JIRA dient als Projektmanagementplattform. Aufgabenmanagement, Fehlerbehebungsprozesse und Statusverfolgung werden von JIRA unterstützt. Einzelne Tickets (Issues) stellen eine Aufgabe oder Teilaufgabe dar und können einzelnen Gruppen oder Personen zugewiesen werden. Über Agile-Boards können Scrum-Prozesse modelliert werden und, wie in Abbildung 3 dargestellt, eine Übersicht über den aktuellen Sprint liefern.

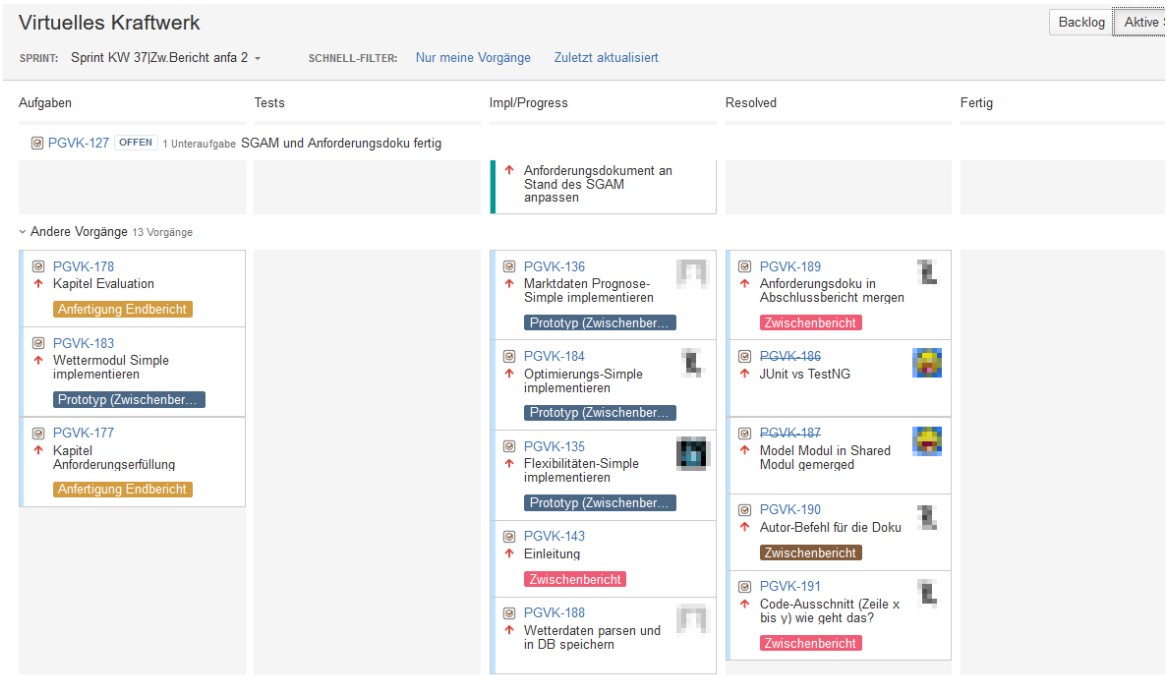


Abbildung 3: Ausschnitt des Sprints aus der Kalenderwoche 37 im projekteigenen JIRA

Für die Zeiterfassung der einzelnen Aufgaben wird das *Tempo Timesheets Plug-in* für JIRA benutzt. Die Zeiterfassung wird übersichtlich pro Aufgabe und Person als Tabelle dargestellt (s. Abb. 4).

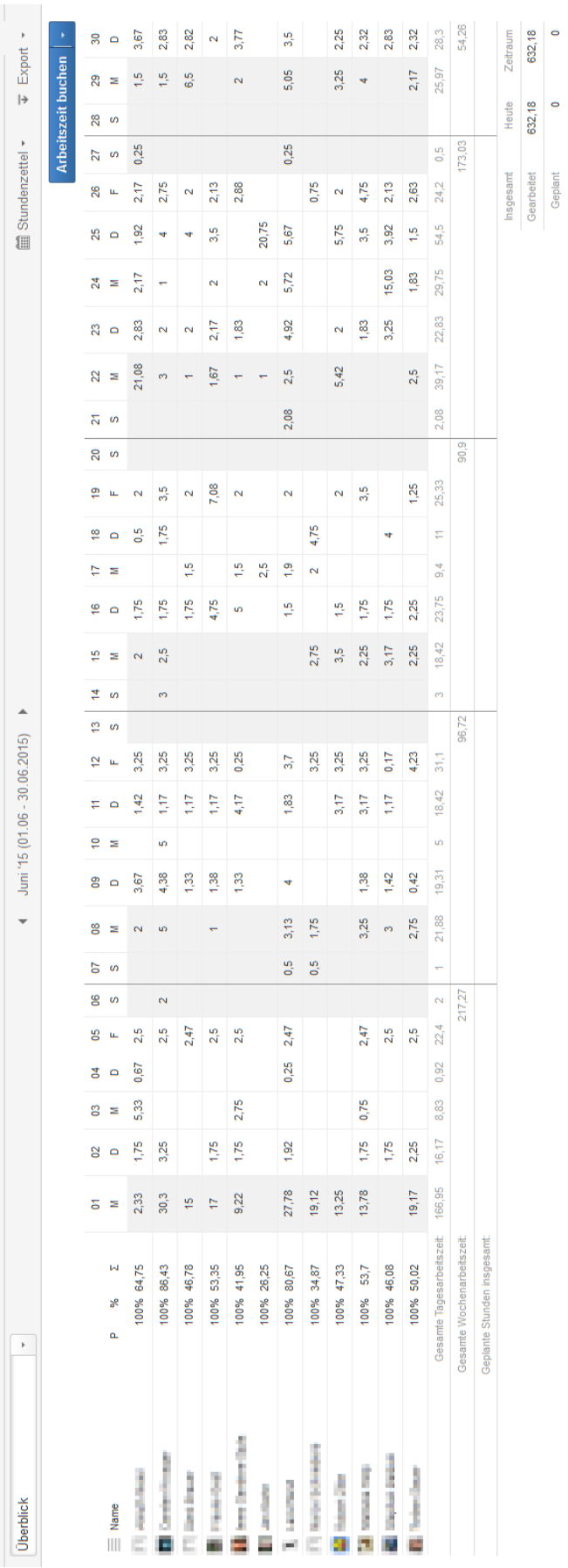


Abbildung 4: Ausschnitt der projekteigenen Zeiterfassung

2.5.3. Etherpad

Die Projektgruppe nutzt ihre eigene Instanz des webbasierten Texteditors Etherpad. Mit diesem können viele Personen an einem Dokument gemeinsam in Echtzeit arbeiten. Es wird zum Verfassen des Protokolls während einer Sitzung und von Kleingruppen für die schnelle und kollaborative Wissenssammlung und Verwaltung von Unteraufgaben genutzt.

Abbildung 5: Eine Tagesordnung im projekteigenen Etherpad

Folgende Werkzeuge machen das Etherpad noch nützlicher:

- Auf der Startseite werden alle Pads mithilfe des Plug-ins *small_list* aufgelistet
- Mithilfe des Plug-ins *table_of_contents* wird rechts dauerhaft ein Inhaltsverzeichnis angezeigt (s. Abbildung 5). Durch Klick auf eine Überschrift scrollt das Dokument an die passende Stelle.
- Das Plug-in *simpletextsearch* stellt eine Suche auf der Startseite bereit, mit der sich alle Pads gleichzeitig durchsuchen lassen.
- Mithilfe eines selbst geschriebenen Python-Skripts wird aus einem Etherpad-Protokoll ein Latex-Dokument erstellt. Dabei werden Überschriften und Baumstrukturen von Stichpunkten übernommen.

2.5.4. Confluence

Zusätzlich zu dem Etherpad wird *Confluence* als Plattform zum permanenten und übersichtlichen Festhalten und für den Austausch von Wissen, Informationen und Empfehlungen genutzt. Vor allem Erklärungen über das Einrichten und Verwenden von Technologien (Kurzanleitungen) werden in Confluence festgehalten. Confluence bietet dabei nicht die Möglichkeit, dass mehrere Personen in Echtzeit an einem Dokument arbeiten können. Stattdessen können Dokumente strukturierter verwaltet und Anhänge beigefügt werden. Die in JIRA benannten Tickets können in die Dokumente von Confluence verlinkt werden, um eine bessere Übersicht über die verbunden Aufgaben zu erhalten.

2.5.5. Slack

Slack ist eine Software, die innerhalb der Projektgruppe zur Kommunikation zwischen den Projektgruppenmitgliedern neben- bzw. zusätzlich zu den Sitzungen verwendet wird. Es gibt einen Webclient, welcher im Browser ohne Installation weiterer Software verwendet werden kann. Zusätzlich

sind Desktop-Anwendungen für Windows, Mac OS X, Linux sowie mobile Anwendungen für Android, iOS und Windows Phone verfügbar. Da der Nachrichtenverlauf online auf den Slack-Servern gespeichert wird, ist dieser entsprechend auf jedem verwendeten Gerät zugänglich.

Slack bietet neben der Möglichkeit direkter Nachrichten zwischen zwei Benutzern auch die Organisation in öffentlichen Kanälen, zu denen alle Gruppenmitglieder Zugang haben, sowie in privaten Gruppen, zu denen nur ausgewählte Mitglieder Zugang haben. Jedes Mitglied kann neue Kanäle und Gruppen erstellen und bei letzteren entscheiden, wer Zugang dazu hat. So organisieren sich Teilgruppen innerhalb der Projektgruppe über Slack-Gruppen beispielsweise zur Terminfindung oder zum weiteren Austausch zwischen den Treffen, während allgemeine Informationen über Kanäle zur Verfügung gestellt werden. Wichtige Informationen, die später schnell wieder abrufbar sein sollen, können entweder von Mitgliedern favorisiert oder auch in Gruppen oder Kanälen angepinnt werden.

Dateien können in Slack komfortabel via *Drag and Drop* hochgeladen werden. Es gibt außerdem die Möglichkeit, via E-Mail Nachrichtenhinweise zu bekommen und dafür einzustellen, ob überhaupt und auf welche Typen von Nachrichten hingewiesen werden soll. Dies kann jeder Benutzer für sich selbst entscheiden. Durch sogenannte *Integrations* (*dt. Funktionsintegrationen*) können außerdem weitere Dienste wie z. B. JIRA, *Bitbucket* oder *Jenkins* integriert werden. Diese Möglichkeit wird innerhalb der Projektgruppe jedoch nicht genutzt.

Innerhalb der Projektgruppe wird Slack in der kostenlosen Version genutzt. Diese beschränkt den Nachrichtenverlauf auf die letzten zehntausend Nachrichten und die Anzahl der Funktionsintegrationen auf zehn ein. Eine Lizenz für die Nutzung von zusätzlichen Funktionen kann durch Bezahlung von aktuell (Stand 03.03.2016) bis zu 15$ pro Nutzer und Monat erworben werden.

2.5.6. LaTeX

Zur Erstellung der Dokumentation wurde das Softwarepaket LaTeX benutzt. Dieses nutzt das Textsatzsystem TeX, welches eine exzellente Formatierung im Blocksatz und insbesondere von mathematischen Formeln und Gleichungssystemen, die sich über mehrere Seiten erstrecken können, ermöglicht. Damit ist es für wissenschaftliche Text prädestiniert. LaTeX ist nach dem „What You See Is What You Mean“-Prinzip aufgebaut, d. h. der Benutzer schreibt seinen Text, aber das Programm kümmert sich um die Formatierung und darum, dass das Ergebnis gut aussieht. Durch zahlreiche Makros und Pakete wird alles ermöglicht[1], was für eine gute Dokumentation benötigt wird. Außerdem kann durch selbst definierte Befehle und Umgebungen eine einfache und konsistente Bearbeitung erlaubt werden.

Für die Speicherung nutzt LaTeX ein Plain-Text-Format. Dadurch wird im Gegensatz zu binären Dateiformaten, wie z. B. bei Microsoft Word, eine einfache Versionierung – in unserem Fall mit Git – möglich. Außerdem besteht die Möglichkeit, das Dokument in mehrere Dateien aufzuspalten und so eine parallele, konfliktarme Bearbeitung zu ermöglichen. Daher wird jedes Kapitel und Unterkapitel in einer eigenen tex-Datei gespeichert.

[1] LaTeX ist Turing-vollständig ☺

2.5.7. Evaluation der Infrastruktur

Die folgende Evaluation der Technologien beurteilt den Einsatz der zuvor vorgestellten, verwendeten Technologien in dem jeweiligen Zusammenhang bzw. Umfeld, in dem sie eingesetzt wurden.

Gruppenkommunikation Zur gruppeninternen Kommunikation wurde das Chat-Programm Slack verwendet. Die einfach zu erstellenden Gruppenkonversationen und die schnelle Kommunikationsmöglichkeit mit Einzelnen verbesserten die Absprache untereinander entscheidend. Ein kleinerer Makel ist das Verlorengehen des Nachrichtenverlaufs mit der Zeit in der kostenfreien Version. Das Programm startet zudem etwas langsam.

Zusätzlich wurden zwei E-Mail-Verteiler eingerichtet: ein Verteiler für die Projektgruppe intern sowie ein Verteiler für die Projektgruppe einschließlich der Betreuer. Dies erwies sich als sinnvoll, um wichtige Termine und Daten auf einem weiteren Weg zu erhalten und ebenso externe Absprachen mit Informations- und Datenquellen weiterzuleiten.

Tagesordnung und Protokoll Zum Protokollieren der Sitzungen wird Etherpad benutzt, ein Webservice, der es erlaubt, mit vielen Personen gleichzeitig an einer Textdatei zu schreiben. Dies erlaubte es Tagesordnungspunkte nach Bedarf auch während der Sitzung anzupassen. Zusätzlich ermöglichte es, dass mehrere Personen kollaborativ protokollieren konnten und somit der Druck auf einen einzelnen Protokollanten genommen wird. Im Anschluss an die Sitzung wird mithilfe eines Python-Skriptes der rohe Text in LaTeXübersetzt und kann mit ein wenig Verbesserung des automatisch erzeugten Dokumentes direkt als Protokoll versendet und versioniert werden. Dieses Vorgehen erwies sich als außerordentlich sinnvoll und wurde sehr positiv bewertet.

Wissen erarbeiten und festhalten Auch bei dieser Tätigkeit wurde wieder Etherpad verwendet. Das dadurch ermöglichte parallele Arbeiten und bequeme erste Textverfassen erleichterte den notwendigen Aufwand. Als negativ erwies sich jedoch bei dieser Arbeit, dass weder Bilder einzufügen noch größere Formatierungen, außer der Einteilung in Überschriften, Auflistungen sowie einfache Textformatierung möglich war. Daher ist es schwierig, etwas zu finden, ohne genaueres Wissen darüber, wo dies zu finden ist. Die Dokumente sind zusätzlich nur als alphabetische Liste strukturiert. Trotzdem war es sinnvoll aufgrund der kollaborativen Erstellung vieler Arbeitsdokumente, diese zunächst im Etherpad zu verfassen oder zumindest vorzustrukturieren.

Confluence hat den Vorteil einer Baumstruktur mit der Einbindung von Bildern, Tabellen, Codeausschnitten und vielem mehr. Zusätzlich kann die komplette Struktur durchsucht werden. Der einzige Nachteil ist die etwas langsame und umständliche Erstellung und Navigation. Auch die gemeinsame gleichzeitige Arbeit an einer Seite ist nicht möglich. Daher wurde Confluence als Wissensdatenbank genutzt, in die bereits erarbeitetes Wissen sowie Anleitungen übernommen werden.

Somit haben sowohl Etherpad als auch Confluence eine Daseinsberechtigung im Projekt.

Stand von Aufgaben festhalten Aus den bereits oben aufgezeigten Gründen ergibt sich, dass es auf Dauer unübersichtlich wäre, Tasks im Etherpad festzuhalten und weiterzuentwickeln. Ohne eine

Suche würde dies sehr schnell unübersichtlich. Daher wird JIRA genutzt, welches es mittels übersichtlicher Spalten eines Agile Boards ermöglicht, den Status einzelner Aufgaben zu kennzeichnen. Zusätzlich ist einfach erkennbar, welche Aufgaben noch übernommen werden können und welche einem selbst zugewiesen wurden. Allerdings sind zu diesen großen Vorteilen auch einige Nachteile hinzuzufügen, die der Projektgruppe in der Evaluation besonders auffielen. Die selbe Aufgabe kann nicht gleichzeitig von mehreren Nutzern geändert werden und das übersichtliche Board lässt sich nur durch Neuladen der Seite aktualisieren. Die Anzahl der notwendigen Klicks um etwas zu erreichen ist subjektiv zu hoch, ebenso wie die Seite selbst relativ langsam ist. Der zusätzliche doppelte Aufwand, der notwendig ist, um die Aufgaben und deren Status ins Protokoll zu übernehmen, führt zur Verlangsamung der Sitzung und der Notwendigkeit jeden einzelnen Task zu überprüfen.

Schlussendlich ist JIRA in seiner Funktionsweise jedoch nicht durch ein für die Verhältnisse der Projektgruppe passenderes System ersetzbar. Die aufgetretenen Probleme bestehen ebenso bei alternativen Werkzeugen.

Stundenzettel Zum Festhalten der Arbeitszeit wird ein Plug-in für JIRA namens *Tempo* verwendet. Dieses erlaubt das Protokollieren der Arbeitszeit, die für eine Aufgabe aufgewendet wurde. Es übernimmt das Berechnen der jeweils komplett aufgewendeten Zeit in einem beliebigen Zeitrahmen und besticht durch die Integration im JIRA. Nachteile bestehen in der Notwendigkeit, über die ohnehin relativ langsame JIRA Seite zusätzliche Arbeit aufbringen zu müssen, um die Zeit einzutragen, die man gearbeitet hat. Es ist weiterhin schwierig Zeit für nebenläufige Arbeit einzutragen, die entsteht, aber nicht direkt mit einer spezifischen Aufgabe in Verbindung steht.

Im Ganzen betrachtet sind diese Nachteile jedoch weniger relevant und das Festhalten der Arbeitszeit geschieht übersichtlich und an einer zentralen Stelle.

Dokumentation Die Einrichtung größerer LaTeX-Schriftstücke ist komplex, so dass die bestehende Erfahrung einiger Gruppenmitglieder damit sehr hilfreich war. Wie bei LaTeXgewohnt, traten gelegentlich kleinere Formatierungsfehler auf, welche aber in keinem Vergleich zur Unhandlichkeit von Microsoft Word bei komplexen Dokumenten stehen. Eine Alternative zu LaTeXsieht die Projektgruppe besonders beim Aspekt der Mächtigkeit nicht.

Schlussfolgerung Die Projektgruppe selbst hat während der Arbeit viele neue und alte Werkzeuge und Technologien benutzt und in ihren Arbeitsfluss erfolgreich integriert. Die resultierende Praxis ist durch die Werkzeuge an verschiedensten Stellen unterstützt und behindert wenn überhaupt, nur unwesentlich und niemals so stark, dass es besser gewesen wäre, dieses spezielle Werkzeug nicht zu verwenden.

3. Vision und Related Work

Für die erfolgreiche Umsetzung des Projekts wurde eine Vision erstellt. Sie definiert zeit- und situationsunabhängig die strategische Ausrichtung des Projekts [Pro]. Die Vision unterliegt zwar nicht den strengen Anforderungen einer Zieldefinition, bietet jedoch einen Blick auf das gewünschte Ergebnis nach der Fertigstellung des Projekts. Neben der Vision (s. Abschnitt 3.1) wurde vor der Erstellung von *Powder* eine Recherche über bereits existierende, ähnliche Systeme oder Teilsysteme durchgeführt (s. Abschnitt 3.2). Diese beiden Bereiche dienen als ein Teil der Grundlagen dazu, *Powder* zu entwickeln.

3.1. Vision

Damit die gegebene Problemstellung gelöst werden kann, muss das System die im Folgenden beschriebenen Funktionen bereitstellen. Zunächst wird aus den Markt- und Wetterdaten eine Prognose der Marktpreise generiert. Außerdem muss die Flexibilität der energieerzeugenden Anlagen in Form ihrer jeweils ausführbaren Fahrpläne auf Basis der Wetterprognose ermittelt werden. Mit diesen Informationen kann ein optimiertes Portfolio erstellt werden. Für dieses Portfolio ist die Berechnung eines optimalen Einsatzplanes erforderlich, damit die Auslieferung der gewählten Produkte garantiert werden kann.

Die Abbildung 6 zeigt das zu entwickelnde System eingebettet in den Systemkontext, in dem es mit nicht systeminternen Komponenten wie der Strombörse, realen Anlagen und verschiedenen Datenquellen in Verbindung gebracht ist. Zu beachten ist, dass die tatsächliche Ausführung und Übergabe der Fahrpläne an reale Anlagen inklusive möglicher reaktiver Steuerung nicht Teil dieses Projekts ist. Weiterhin ist der simulierte Handel des erstellten Portfolios am Markt ebenfalls nicht vorgesehen, da die notwendige technische Schnittstelle für dieses Projekt nicht zur Verfügung steht.

3.2. Related Work

In diesem Abschnitt werden die relevantesten Forschungsarbeiten aufgeführt, die sich mit demselben Thema befasst haben, wie die Projektgruppe. Dabei wird auch die überlegte Umsetzung, die in der Vision 3.1 dargestellt wird, berücksichtigt. Ziel dieses Kapitels ist die Identifizierung von relevanten Ansätzen, die für die Projektgruppe von Interesse sein könnten. Bei der ersten Recherche fiel jedoch auf, dass es keine Forschungsgruppe mit einer fast identischen Problemstellung gibt, sodass auch die möglichen Umsetzungen schwer miteinander verglichen werden können. Dadurch ist es auch nicht möglich, direkte Anhaltspunkte für die eigene Umsetzung daraus zu gewinnen. Es wurde sich daher auf ähnliche Problemstellungen und deren Ansätze konzentriert, welche einen großen Teil der Aufgaben umgesetzt haben, die auch im Projekt bearbeitet wurden. Dabei wurden die folgenden vier Ansätze genauer betrachtet.

Mittelfristige risikoorientierte Optimierung von Strombeschaffungsportfolios kleinerer Marktteilnehmer: Die erste relevante Arbeit [GKLL+03] mit einer ähnlichen Problemstellung befasst sich mit der Portfolioerstellung eines Markteilnehmers mit physischen Anlagen zur Eigenenergienutzung. Dabei wird auch auf finanzielle Instrumente an der Strombörse und die Prognose verschiedener

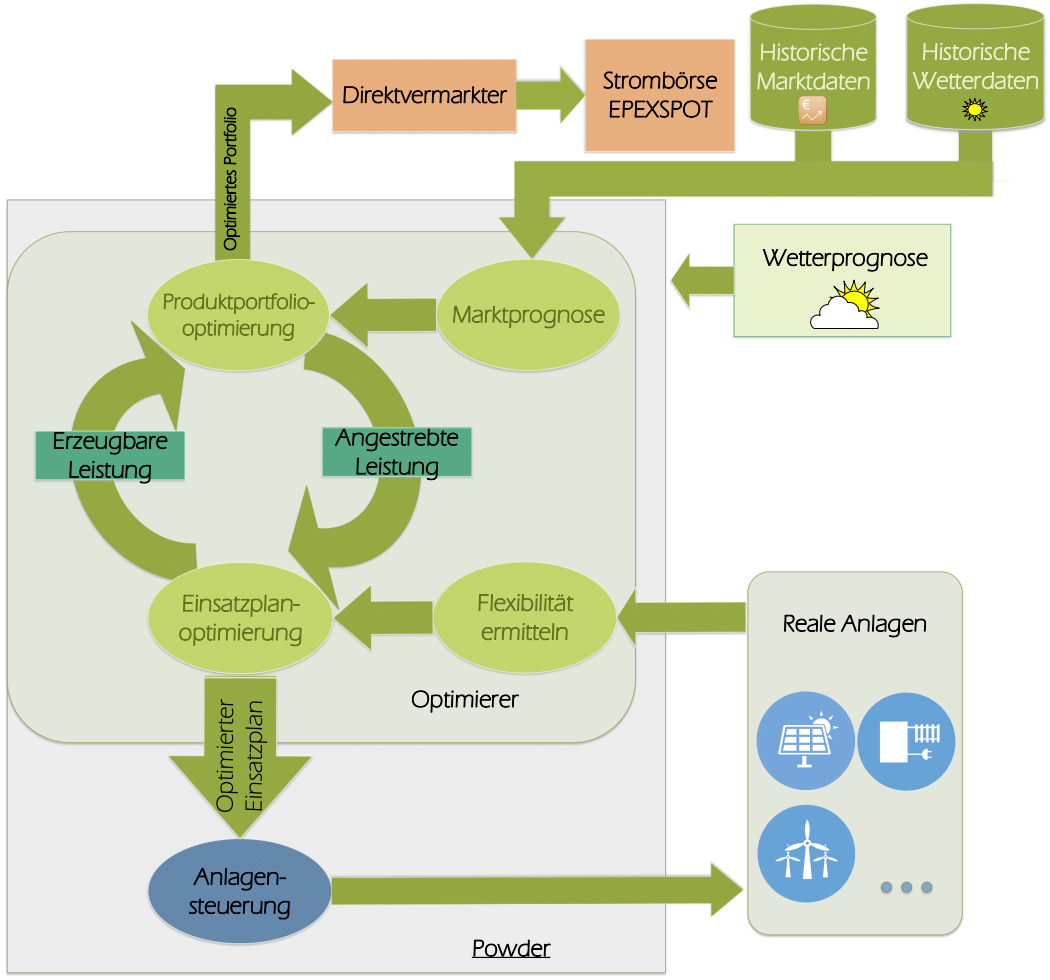


Abbildung 6: Visionsgrafik der PG VK

Modellparameter zur Deckung der entstandenen Last eingegangen. Ziel der Arbeit ist die Maximierung des Portfoliowerts unter der Berücksichtigung des Risikos. Dazu werden unter anderem die Temperatur als größtes Prognosemerkmal für die Lastberechnung eingesetzt und eine Normierung der Tage für die Prognose vorgenommen.

Da jedoch die Maximierung unter dem Gesichtspunkt der Lastabdeckung durchgeführt wird und auch der Eigenbedarf eine übergeordnete Rolle spielt, kann dieser Ansatz und die entsprechende Problemstellung nicht direkt auf die Aufgabe der Projektgruppe abgebildet werden.

Optimierung eines Portfolios mit hydrothermischem Kraftwerkspark im börslichen Strom- und Gasterminmarkt: Die zweite Arbeit [Bag02] wurde an der Universität Stuttgart erstellt. Bei dieser handelt es sich um einen Forschungsbericht über die Optimierung eines Portfolios mit hydrothermischen Kraftwerken für den börslichen Strom- und Gasmarkt. In diesem Zusammenhang wird eine Methode entwickelt, mit der ein Optimum im Geschäftsportfolio eines konzerneigenen Kraftwerksparks erzeugt werden soll. Es werden Restriktionen der einzelnen Anlagen sowie die Marktpreisbewegung an der Strombörse (EEX) berücksichtigt. Dazu werden problemspezifische Algorith-

men wie beispielsweise die *Langrange Relaxation* verwendet, um die hohen Rechenzeiten für die großen Planungshorizonte zu verringern. Für die flexible Anpassung beim Auftreten von Veränderungen an der Strombörse, neuen Produkten oder Vertragsformen, wurde der Lösungsalgorithmus *Branch & Bound* eingesetzt, wodurch die Rechenzeit auf wenige Minuten reduziert wurde.

Für den Ansatz der Projektgruppe ist vor allem der Umgang mit den hohen Rechenzeiten und die Einhaltung der Vertragsformen an der Börse von Interesse. Da in dieser Arbeit Anlagentypen verwendet werden, die für die Bearbeitung der Aufgabe der Projektgruppe nicht vorgesehen sind, ist dieser Ansatz jedoch nur in einzelnen Bereichen von Interesse. Die Arbeit wurde im Jahr 2002 angefertigt, in welchem die Direktvermarktung von EE-Anlagen noch nicht eingeführt worden war, sodass der Handel an der Börse nicht mit der heutigen Komplexität vergleichbar ist und somit einen wichtigen Aufgabenteil der Projektgruppe nicht abdeckt.

Entwicklung und Anwendung einer Kraftwerks- und Speichereinsatzoptimierung für die Untersuchung von Energieversorgungsszenarien mit hohem Anteil erneuerbarer Energien in Deutschland: Wie bereits der Titel darstellt, wird in dieser Arbeit [vO12] von der Universität Kassel versucht, eine Anwendung zu entwickeln, mit der Kraftwerke und Speicher so optimiert werden können, dass die Energieversorgung Deutschlands im Jahre 2050 gesichert ist. Die Optimierung bezieht sich auf die Einsatzpläne der einzelnen Anlagen. In der Arbeit wurde das Optimierungsproblem mit einem linearen *Branch-and-Cut-Solver* gelöst.

Für die Projektgruppe war vor allem die Herangehensweise an das Optimierungsproblem der Einsatzplanung, sowie die Modellierung der Kraft Wärme Kopplungsanlagen (KWK Anlagen) von Interesse. Da jedoch auch in diesem Ansatz nur eines der Optimierungsprobleme bearbeitet und in diesem Fall die wirtschaftliche Optimierung an der Börse nicht betrachtet wurde, konnte auch dieser Ansatz nur geringfügig für die Bearbeitung der Problemstellung der Projektgruppe genutzt werden.

Smart Nord: Als letzter Ansatz wurde das Projekt *Smart Nord* [hLHrnhMS15] der Universität Oldenburg betrachtet. In dem Projekt wurde in Kooperation mit der Universität Hannover im Bereich der erneuerbaren Energien geforscht. Die Forschungsarbeit unterteilt sich in sechs Bereiche:

1. DVPP (dynamic virtual power plants)
2. Netzstabilisierende Systemdienstleistungen
3. Integrierter Markt
4. Verteilungs- und Übertragungssystem
5. Systemtheorie (Beschreibung Erklärung von grundlegenden Aspekten Prinzipien) für aktive Verteilnetze
6. Smart Spatial (Quasi intelligente räumliche Verteilung)

Für die Projektgruppe waren vor allem die ersten beiden und teilweise der dritte Bereich von Interesse. Der erste Bereich handelt von der Erstellung eines virtuellen Kraftwerks, in dem die Energieerzeuger

als Agenten aufgefasst werden. Durch ein Multiagentensystem wurden die Energieerzeuger selbstorganisierend dazu gebracht, Koalitionen zu bilden und ein Produktportfolio für den Day-Ahead Markt zu erstellen. Des Weiteren wurde für die Optimierung auch der Intraday Markt für Nachbesserungen hinzugezogen. Für die Projektgruppe konnte vor allem die Herangehensweise mit beiden Optimierungsproblemen genutzt werden. Zudem wurde auch die Lösung durch ein Multiagentensystem überlegt. Im zweiten Bereich, der sich mit der Bereitstellung von netzstabilisierenden Systemleistungen durch erneuerbare Energieanlagen beschäftigt, waren lediglich die verwendeten Kommunikationsprotokolle und Technologien zu den Anlagen, sowie das Vorgehen zur Umsetzung in diesem Bereich von Interesse. Im dritten Bereich geht es um die Überprüfung eines Marktes, in dem mit einer hohen Anzahl von fluktuierenden Erzeugern gehandelt werden soll.

Für die Projektgruppe konnte dieser Bereich jedoch nur als Informationsquelle für Marktmechanismen und den Einsatz von Methoden zur Netzstabilisation verwendet werden. Bei der Recherche zeigte sich, dass das Forschungsprojekt *Smart Nord* die ähnlichsten Themenüberschneidungen mit der Aufgabe der Projektgruppe besitzt und somit dem Ansatz das meiste Interesse und eine sorgfältigere Recherche durch die Projektgruppe zugekommen ist. Da die Projektgruppe in ihrer Umsetzung jedoch kein Multiagentensystem verwenden will, konnte auch dieser Ansatz nur geringfügig als Informationsquelle dienen.

Zusammenfassend zeigt sich, dass es wenig ähnliche Ansätze gibt und diese sich größtenteils von dem der Projektgruppe unterscheiden. Daher wurde die Recherche auf Ansätze für bestimmte Bereiche des Projektes ausgeweitet, sodass für die einzelnen Bereiche jeweils Related Work betrachtet wurde. Die Ergebnisse dieser Recherchen wurden aus Gründen des Leseflusses auch in die dafür vorgesehen Kapitel des Bereiches platziert, wie beispielsweise Marktprognose 6.7, Optimierer 6.9 oder die Auswahl der Kommunikationsstandards 6.2.

4. Anforderungen

In diesem Kapitel werden die Anforderungen an das System erläutert. Sie wurden zunächst durch Interviews mit den PG-Betreuern erarbeitet und in den spezifischen Anforderungen festgehalten. Die Nähe des Projekts zum Energiesektor legte die Verwendung von SGAM zur Erhebung weiterer funktionaler Anforderungen nahe. Daher untergliedert sich dieses Kapitel in eine detaillierte Beschreibung des Anforderungsvorgehens, den spezifischen Anforderungen, sowie SGAM Analyse- und Architekturphase.

4.1. Vorgehen in der Anforderungsanalyse

Die Anforderungserhebung bestand im Wesentlichen aus zwei Schritten: Zunächst führten einige Projektgruppenmitglieder Interviews mit den Projektbetreuern und leiteten daraus die *spezifischen Anforderungen* ab. Im zweiten Schritt wurde SGAM eingesetzt, um die funktionalen Anforderungen zu definieren und eine grobe Architektur zu entwickeln. Diese grobe Architektur ist dennoch Teil des Anforderungskapitels, da es nicht vollständig modelliert ist und nur zur groben Strukturierung und Übersicht von *Powder* dienen soll.

Bei SGAM handelt es sich um eine dreidimensionale Referenzarchitektur, die aus den fünf *Interoperabilitätsebenen* besteht und dabei einem Top-Down-Vorgehen ähnelt. Die oberen zwei Ebenen zählen deshalb zur *Analysephase*, die unteren zur *Architekturphase* [Neu14b, 4-12]. Jede Ebene besteht aus *Domänen* und *Zonen*. Domänen stellen die Energieumwandlungskette dar und Zonen entsprechen den hierarchische Ebenen der Netzführung. Die Einteilung in Domänen und Zonen ermöglicht eine frühe Erkennung von Interoperabilitätsprobleme [Nie15, 16]. Eine genaue Erläuterung des SGAM war Teil einer Seminararbeit und kann im Anhang im Kapitel 7 unter G nachgelesen werden.

Grundsätzlich kann SGAM losgelöst von Anwendungssoftware verwendet werden. Zur einfachen Skizzierung kann das Aufzeichnen mit Stift und Papier oder einer Microsoft Powerpoint-Vorlage ausreichen. Für tiefere Analysen ist jedoch die Benutzung des *Enterprise Architekt* mit dem *SGAM Toolbox*-Plug-in ratsam. Die Installationsdateien sind frei verfügbar auf der SGAM-Toolbox Webseite[2]. Installation und Verwendung der Toolbox kann dem Handbuch [Neu14b] entnommen werden.

Das Handbuch bietet einen Vorschlag zur Bearbeitungsreihenfolge der SGAM-Ebenen, der auch in diesem Projekt zu einem Großteil benutzt wurde. Ein wesentlicher Unterschied ist, dass die Smart Grid Plane (SG-Plane) in vielen Ebenen zunächst ausgelassen und dann erst am Schluss der Bearbeitungsreihenfolge erstellt wurden.

Im Rahmen der Analysephase wurde außerdem ein Workshop mit den Projektbetreuern durchgeführt und auf den Ergebnissen aufbauend ein projektgruppenmitgliederinternes Brainstorming durchgeführt. Anschließend bearbeiteten die Mitglieder der Projektgruppe paarweise die Ergebnisse des Brainstormings.

[2] http://www.en-trust.at/downloads/sgam-toolbox/

4.2. Stakeholder

Stakeholder sind all jene internen oder externen Personen oder Gruppen, die ein berechtigtes Interesse am Verlauf oder Ergebnis des Projekts haben. Zudem sind diese in den meisten Projekten direkt in dem Ablauf mit einbezogen. Für die Identifizierung der Stakeholder im Rahmen der Projektgruppe muss zwischen zwei Arten von Stakeholdern unterschieden werden.

Die *projektbezogenen* Stakeholder haben dabei ein Interesse und somit auch Anforderungen an das Projekt selbst. Dies kann den Umfang, Ausführung und auch verwendete Werkzeuge beinhalten. Des Weiteren können diese auch den Ablauf des Projektes und deren Ziele mitgestalten. In Bezug auf die Projektgruppe handelt es sich bei diesen Stakeholdern sowohl um die Projektmitglieder, als auch um die Projektbetreuer, die sowohl als Kunden bzw. Auftragsstelle und auch als Berater agieren. Dies liegt an der Definition des Projektes. Durch das Mitspracherecht jedes einzelnen Projektmitglieds bei der Umsetzung des zu entwickelnden Systems und dem Ziel, eine möglichst gute Bewertung zu erhalten, besitzt auch jeder einzelne von ihnen Anforderungen, die mit in das Projekt eingebracht werden. Dadurch können diese auch als Stakeholder definiert werden. Da es sich um ein Universitätsprojekt ohne Beteiligung weiterer Firmen handelt, sind die Projektbetreuer und Projektgruppenmitglieder auch die einzigen projektbezogenen Stakeholder, die beachtet werden müssen.

Bei den *systembezogenen* Stakeholdern handelt es sich um die Stakeholder, die direkt mit dem zu entwickelnden System in Verbindung stehen. Dabei handelt es sich zum einen um den Produktabnehmer. Im Falle der Projektgruppe ist dies der Direktvermarkter, der das Bedürfnis besitzt, ein optimales Produktportfolio zu erhalten, welches für ihn verständlich ist und mit welchem dieser an der Börse handeln kann. Dazu kommen die Anlagenbetreiber, die ihre Anlagen dem VK-Betreiber zur Verfügung stellen. Bei diesem besteht eine Reihe von Anforderungen auf beiden Seiten, wodurch es sich um einen zentralen und wichtigen Stakeholder handelt. Als weitere systembezogene Stakeholder können die Marktdatenprovider und Wetterdatenprovider genannt werden. Diese besitzen Anforderungen in Bezug auf die Verwendung ihrer bereitgestellten Daten, wie z. B. Vertraulichkeit, die im Projekt beachtet werden müssen. Als letzter Stakeholder kann der VK-Betreiber gesehen werden, welcher bestimmte Daten, Informationen und Schnittstellen benötigt, um das zu entwickelnde System auch ausführen zu können. Die Anforderungen, die durch die einzelnen Stakeholder an das Projekt entstehen, werden in der Analysephase und in dem Abschnitt der speziellen Anforderung 4.3.1 dargestellt.

4.3. Spezifische Anforderungen

Auf Basis der Interviews mit den Projektbetreuern (s. 4.1) und spezieller Anforderungen der zuvor identifizierten Stakeholder wurden die spezifischen Anforderungen für das Projekt erstellt. Dabei handelt es sich um alle Anforderungen an das Projekt, die nicht direkt an das zu entwickelnde System gerichtet sind, wie beispielsweise die Vertraulichkeit von Daten oder welche Anlagentypen verwendet werden sollen. Aus den spezifischen Anlagen und dem SGAM-Modell werden dann in den folgenden Abschnitten die systemrelevanten Anforderungen (funktionale Anforderungen) entwickelt 4.1.

Für die Übersichtlichkeit der spezifischen Anforderungen wurden diese in sechs Kategorien unterteilt:

- Leistungsanforderungen

- Schnittstellenanforderungen
- Datenbankanforderungen
- Entwurfsanforderungen
- Qualitätsanforderungen
- Weitere Anforderungen

Im Folgenden werden nun die einzelnen Anforderungen jeder Kategorie aufgeführt.

4.3.1. Leistungsanforderungen

Die erste Kategorie der Anforderungen sind die Leistungsanforderungen. Hierbei handelt es sich um die Leistungen, die das fertige Produkt beinhalten soll, sowie auch die Leistungen, die nicht in der Umsetzung betrachtet werden sollten. Aus den Interviews und den Sitzungen konnte entnommen werden, dass für das Anlagenportfolio nur eine begrenzte Anzahl an unterschiedlichen Anlagentypen verwendet werden soll. Es wurde festgelegt, dass die Anlagentypen Photovoltaik-Anlage (PV-Anlage) und Blockheizkraftwerk (BHKW) integriert werden sollen (Anforderungen L-01 und L-02). Als zusätzlicher Anlagentyp, der gegebenenfalls integriert werden könnte, wurde eine stationäre Batterie bestimmt (Anf. L-03). Für das Erstellen der Anlagenportfolios soll jedoch das dynamische Hinzufügen und Entfernen von Anlagen dieser Anlagentypen (parametrisierte Anlagentypen) möglich sein (Anforderungen L-04 und L-05). Des Weiteren wurde von den Stakeholdern beschlossen, dass keine Verbraucher in die Betrachtung der Optimierung des Produktportfolios miteinbezogen werden müssen (Anf. L-06).

Bei der Anlagenmodellierung sowie der Optimierung für das Produktportfolio wurden die Anforderungen beschlossen, dass die Betriebskosten der Anlagen berücksichtigt werden sollen (Anf. L-08), aber nicht die Anschaffungskosten (Anf. L-07). Für die Berechnung der Betriebskosten des BHKWs sollen die Taktung während des Betriebs (Anf. L-09), die Abkühlung der Temperatur innerhalb des Pufferspeichers (Anf. L-10) und die Kosten für das Hoch- und Herunterfahren des BHKWs mit einbezogen werden (Anf. L-11). Für die Berechnung soll auf die Werte von realen BHKWs eines frei auswählbaren Herstellers zurückgegriffen werden. Auch die Parameterkonfigurationen eines BHKWs oder auch einer PV-Anlage, wie beispielsweise maximale Leistung, sollen den Herstellerangaben entnommen werden (Anf. L-12). Als weitere Restriktion neben den Betriebskosten soll für das BHKW eine wärmegeführte Betriebsführung angenommen werden (Anf. L-13). Dabei soll das BHKW nur verwendet werden, um den Wärmebedarf eines typischen Einfamilienhaushalts direkt, oder über das Füllen des Speichers, zu decken (Anf. L-14). Für den Wärmebedarf eines Einfamilienhaushalts soll dafür auf Standardlastprofile zurückgegriffen werden.

Des Weiteren wurde vereinbart, alle Berechnungen, die für die Optimierung nötig sind, auf Basis von historischen Datensätzen durchzuführen (Anf. L-15). Auch wurde vereinbart, dass der Direktvermarkter als Schnittstelle zur Börse gilt und nicht als Stakeholder. Dieser kauft entweder das erstellte Produkt oder nicht. Er hat also kein Mitspracherecht, wie die Gestaltung des Produktes aussehen soll (Anf. L-16). Auch das Mitspracherecht des Anlagenbetreibers soll sehr stark vereinfacht dargestellt

werden. Für das Projekt reicht es dabei aus, dass dieser das Ziel eines maximalen Gewinns besitzt. Dafür dürfen die bereitgestellten Anlagen so geregelt werden, wie es das Produktportfolio beschreibt (Anf. L-17). Deshalb muss der Anlagenbetreiber auch nicht als Stakeholder angesehen werden, sondern lediglich die Produktabnehmer.

Für die Modellierung der Simulationen wurde vereinbart, dass die Anlagen nicht auf Echtzeitbasis simuliert werden müssen (Anf. L-18), sowie dass es sich um eine Simulation der Anlagen handeln darf und nicht um physische Anlagen selbst (Anf. L-19).

L-01 Das System muss PV-Anlagen als Anlagentyp unterstützen. (umgesetzt, s. S. 143)

L-02 Das System muss Blockheizkraftwerk-Anlagen als Anlagentyp unterstützen. (umgesetzt, s. S. 148)

L-03 Das System kann stationäre Batterien als Anlagentyp unterstützen. (verworfen, s. S. 205)

L-04 Das System muss das dynamische Hinzufügen von parametrisierten Anlagentypen unterstützen. (umgesetzt, s. S. 113)

L-05 Das System muss das dynamische Entfernen von parametrisierten Anlagentypen unterstützen. (umgesetzt, s. S. 113)

L-06 Das System muss bei der Portfoliooptimierung keine Verbraucher einbeziehen. (umgesetzt, s. S. 127)

L-07 Das System muss bei der Portfoliooptimierung keine Anschaffungskosten der Anlagen berücksichtigen. (umgesetzt, s. S. 144)

L-08 Das System muss bei der Portfoliooptimierung die Betriebskosten der Anlagen berücksichtigen. (umgesetzt, s. S. 148)

L-09 Das System sollte bei der Berechnung der Betriebskosten der Anlagen die Taktung während des Betriebs einbeziehen. (umgesetzt, s. S. 151)

L-10 Das System sollte bei der Berechnung der Betriebskosten die Abkühlung des Pufferspeichers einbeziehen. (umgesetzt, s. S. 145)

L-11 Das System sollte bei der Berechnung der Betriebskosten die Kosten für das Hoch- und Herunterfahren des Blockheizkraftwerks einbeziehen. (verworfen, s. S. 148)

L-12 Das System muss auf Grundlage realer Konfigurationen der Anlagenparameter von einem beliebigen Herstellers arbeiten. (umgesetzt, s. S. 113)

L-13 Das System muss für die Blockheizkraftwerke eine wärmegeführte Betriebsführung annehmen. (umgesetzt, s. S. 114)

L-14 Der wärmegeführten Betriebsführung soll der typische Wärmebedarf eines Einfamilienhaushalts zu Grunde gelegt werden. (umgesetzt, s. S. 114)

L-15 Das System sollte historische Daten für die Optimierung verwenden. (umgesetzt, s. S. 126)

L-16 Das System muss das Portfolio selbstständig zusammenstellen. (umgesetzt, s. S. 159)

L-17 Das System muss als Ziel des Anlagenbetreibers den maximalen Gewinn berücksichtigen. (umgesetzt, s. S. 154)

L-18 Die Anlagensimulationen könnte echtzeitfähig sein. (verworfen, s. S. 132)

L-19 Die Anlagen müssen nicht physisch vorhanden sein, sondern können auf einer beliebigen Maschine simuliert werden. (umgesetzt, s. S. 128)

4.3.2. Schnittstellenanforderungen

Neben den speziellen Leistungsanforderungen wurden auch Anforderungen an Schnittstellen durch die Stakeholder gestellt. Dabei wird von diesen gefordert, dort wo es möglich ist, standardbasiert zu kommunizieren (Anf. S-01). Darüber hinaus sollen mindestens drei technische Schnittstellen implementiert werden. Dabei soll es sich um Schnittstellen zu den Anlagen und zu den Datenbanken oder der Datenbank für die historischen Markt- und Wetterdaten handeln (Anforderungen S-02, S-03 und S-04).

S-01 Das System sollte Standards für die Kommunikation verwenden, wenn Standards vorhanden sind. (umgesetzt, s. S. 113)

S-02 Das System muss eine Schnittstelle zu den Anlagen besitzen. (umgesetzt, s. S. 112)

S-03 Das System muss eine Schnittstelle zur externen Datenbank der historischen Marktdaten besitzen. (umgesetzt, s. S. 112)

S-04 Das System muss eine Schnittstelle zur externen Datenbank der historischen Wetterdaten besitzen. (umgesetzt, s. S. 118)

4.3.3. Datenbank-Anforderungen

Für die Datenbank wurde keine spezielle Anforderung gestellt. Es wurde lediglich vereinbart, die Datenbank als nichtmenschlichen Akteur in das SGAM aufzunehmen.

4.3.4. Entwurfsanforderungen

Für den Entwurf des Systems wurde vereinbart, dass das System ein virtuelles Kraftwerk enthält (Anf. E-01). Wie die Umsetzung des virtuellen Kraftwerks und der weiteren Module aussieht, ist der Projektgruppe aber selbst überlassen. Gerade bei den Optimierungsalgorithmen darf die Projektgruppe frei entscheiden, solange die restlichen Anforderungen eingehalten werden.

E-01 Das System muss ein virtuelles Kraftwerk umsetzen. (umgesetzt, s. S. 111)

4.3.5. Qualitätsanforderungen

Als letzter Unterpunkt der speziellen Anforderung sollen die Qualitätsanforderungen betrachtet werden. Dabei wurden keine expliziten Anforderungen seitens der Stakeholder gestellt. Es wurde lediglich gefordert, dass die erstellten Produktportfolios auf ihre Qualität hin evaluiert werden sollen, damit die Güte dieser bestimmt werden kann (Anf. Q-01). Aber auch die Evaluationsmethode darf von der Projektgruppe frei gewählt werden. Weiterhin sollen für die Qualitätsbetrachtung die Algorithmen quantifiziert werden. Da es sich zunächst um ein Forschungsprojekt handelt, müssen auch keine Vertragsstrafen bei Vertragsbrüchen oder der nicht Einhaltung der geforderten Leistung im Projekt berücksichtigt werden (Anf. Q-02).

Q-01 Das System muss die Qualität des erstellten Produktportfolios bestimmen. (umgesetzt, s. S. 172)

Q-02 Das System muss keine Vertragsstrafen berücksichtigen, wenn nicht so viel Energie geliefert wurde, wie vereinbart. (umgesetzt, s. S. 155)

4.3.6. Weitere Anforderungen

Unter weiteren Anforderungen wurden nur die Abgabetermine für die Berichte und das Endprodukt vereinbart.

Damit wurde alle speziellen Anforderungen neben den funktionalen Anforderungen aufgelistet und aufgezeigt. Diese sollen genau wie die funktionalen Anforderungen im Entwurf des Systems mit berücksichtigt werden.

4.4. SGAM: Analysephase

Wie bereits im Abschnitt 4.1 beschrieben, setzt sich das SGAM aus fünf Ebenen zusammen. Die Bearbeitung der ersten beiden Ebenen *Business Layer* und *Function Layer* werden dabei zusammen für die Analysephase und somit der Identifizierung der funktionalen Anforderungen verwendent. Beide Ebenen stellen zusammen ein Top-Down-Modell dar, indem die funktionalen Anforderungen immer weiter spezifiziert werden. Damit die funktionalen Anforderungen so genau wie möglich identifiziert werden können, werden diese innerhalb der beiden Ebenen noch einmal in drei Beschreibungsebenen aufgeteilt. Diese Ebenen werden Business Use Case (BUC), High Level Use Case (HLUC) und Primary Use Case (PUC) genannt. Auf jeder dieser Ebenen werden die einzelnen funktionalen Anforderungen immer weiter auf kleinere Anforderungen aufgeteilt und spezifizierter betrachtet, sowie beschrieben. Ein Vorteil bei diesem Vorgehen ist auch, dass in jeder Phase die Stakeholder einen detaillierten Einblick auf die Anforderungen gewinnen können und auch an der Erstellung dieser mitarbeiten können. Die einzelnen Anforderungen werden im jeweiligen Abschnitt erläutert. Eine genauere Betrachtung des gesamten SGAM kann aus Seminarband G in Kapitel 7 entnommen werden. Als erster Schritt wird der Business Layer für die Erstellung der funktionalen Anforderungen des Projektes durchgeführt.

4.4.1. Business Layer

Die erste Ebene des SGAM und der erste Schritt der Analysephase ist die Ebene *Business Layer*. Diese beschreibt die Geschäftssicht des jeweiligen Energieprojektes in SGAM. Im *Business Layer* werden die wirtschaftlichen Ressourcen und Vorgänge des Prozesses über *Layer* dargestellt. Dazu müssen zunächst die *Business Actor* (*dt.* Geschäftsakteure) des Projektes ermittelt werden. Dabei handelt es sich um die realen, beteiligen Personen oder Gruppen (Stakeholder) des Projektes, welche unterschiedliche *Business Goals* (*dt.* Geschäftsziele) und somit auch Anforderungen an die Erfüllung der Aufgabe besitzen. Nach [Gro13] kann für jeden Geschäftsakteur auf Basis seiner Vorstellungen ein *Business Use Case* (*dt.* Geschäftsanwendungsfall) erstellt werden. Der *Business Use Case* ist somit die erste grobe Beschreibung der funktionalen Anforderungen und an der Spitze des Top-Down-Modells anzusiedeln.

Die Erstellung der Business Use Case (BUC) (BUC) und die Identifizierung der *Business Actor* wurde zusammen mit den Projektbetreuern durchgeführt. Dabei wurden die *Business Actor* und deren *Business Goals* und die *Business Use Cases*, unter zur Hilfenahme spezifischen Anforderung 4.3 und der Vision, in einem Workshop erarbeitet und entworfen. Die Abbildung 67 (Anhang Seite 229) zeigt die aus dem Workshop hervorgegangenen zwei Business-Cases.

Es wurden die BUC *Energie-Produktportfolio optimieren* und *Anlagen bereitstellen* erarbeitet, welche im Weiteren nun genauer beschrieben werden.

BUC *Energie-Portfolio optimieren* Der BUC *Energie-Produktportfolio optimieren* wird von dem *Business Actor* Energiedienstleister mit dem Ziel ausgeführt, einen optimierten Handel am aktiven Markt abzuschließen. Um das Ziel des *Business Cases* zu erreichen wurde ein grober Prozess erarbeitet (s. Abbildung 67), indem zunächst ein virtuelles Kraftwerk gestartet wird. Separat dazu wird eine Wetterprognose erstellt, mit welcher die Flexibilitäten der einzelnen Anlagen erstellt und eine Marktprognose errechnet werden kann. Mit der Erfassung aller Daten kann mit einer Wechselwirkung zwischen Produktportfoliooptimierung und Einsatzplanoptimierung ein optimales Produktportfolio erstellt werden. Um also ein optimiertes Produktportfolio zu erstellen, muss der Prozess mit seinen sechs groben Schritten ausgeführt werden. Der Ablauf wird dabei durch Pfeile gekennzeichnet. Die Schritte werden als HLUC in SGAM bezeichnet, welche später auch noch genauer erläutert werden. Auch diese wurden während des Workshops erarbeitet und genauer spezifiziert.

BUC *Anlagen bereitstellen* Der zweite BUC *Anlagen bereitstellen* ist ein wesentlich kleinerer Anwendungsfall. Bei diesem hat der *Business Actor* das Ziel, die Flexibilitäten an den Energiedienstleister zu verkaufen. Dies wird über einen sogenannten *Flexibility Contract* geregelt, wodurch es dem Energiedienstleister möglich ist, die Anlagen des Anlagenbetreibers so zu regeln, wie es für ihn optimal ist. Der BUC besitzt nur einen Prozessschritt, in dem die Stammdaten der Anlagen bereitgestellt werden sollen. Diese Daten werden später im virtuellen Kraftwerk benötigt, um ein Anlagenportfolio erstellen und Flexibilitäten berechnen zu können. Somit besteht eine Abhängigkeit zwischen den *Business Cases*.

Bei den in der Beschreibung der BUC eingeführten HLUC handelt es sich um eine kurze Beschreibung des Hauptprozesses, in diesem Fall des jeweiligen *Business Cases*. Die HLUC zeigen dabei die groben Prozessschritte des *Business Cases* auf und besitzen dazu noch eine Beschreibung zu jedem dieser Schritte. In dem Top-Down-Modell handelt es sich deshalb um die zweite Stufe, die bereits etwas detaillierter die funktionalen Anforderungen beschreibt. Die Erstellung der HLUC wird im SGAM sowohl auf der Ebene des *Business Layers*, als auch der Ebene des *Function Layers* zugeordnet. Die genauere Beschreibung der HLUC wird deshalb im Abschnitt des *Function Layers* erfolgen, welcher im Anschluss an dieses Kapitel erfolgt. Erstellt wurden die HLUC und deren Beschreibungen im selben Workshop, wie die BUC. Zunächst wird jedoch auch die Ebene des *Function Layers* selbst genauer betrachtet.

4.4.2. Function Layer

Die Ebene des *Function Layers* ist die zweite Ebene im SGAM für die Analysephase. Diese beschreibt die Funktionen, Services und deren beinhalteten Abhängigkeiten aus dem Blickpunkt der Architektur des zu erstellenden Systems. Das bedeutet, dass die Funktionen und Services unabhängig von Akteuren, physischen Implementationen in Applikationen, Systemen und Komponenten betrachtet werden. Diese stellen lediglich die *Use Case*-Funktionalität dar.

Dazu werden im *Function Layer* die HLUC und PUC verwendet. Während die HLUC bereits eingeführt wurden, handelt es sich bei den PUC um eine genauere Beschreibung der HLUC. Dafür werden diese in kleiner Funktionen bzw. Aufgaben unterteilt und noch weiter spezifiziert. Für beide Beschreibungen werden nach dem SGAM-Vorgehen in [Gro13] Aktivitätsdiagramme verwendet.

Die PUC stellen die unterste Ebene des Top-Down-Modells dar und beschreiben die funktionalen Anforderungen ausführlich. Dazu wurden Anforderungstabellen erstellt, in denen Informationen über die Vorbedingungen, Verbindlichkeiten usw. der PUC dargestellt werden. Die grobe Identifizierung der PUC wurde im Anschluss an den Workshop in einer Brainstorming-Runde mit allen Projektmitgliedern durchgeführt. Die genauere Beschreibung jedes einzelnen PUC wurden danach von Gruppen mit zwei Personen erstellt. Diese wurde jeweils einen HLUC und deren PUC zugeteilt und wurden von diesen auch bearbeitet. Im weiteren Verlauf des SGAM haben diese beiden Personen ihren HLUC in jeder anderen Ebene weiter bearbeitet (s. Vorgehen 4.1). Außerdem beinhaltet jede Tabelle eine Referenz zu dem Abschnitt in dem die Beschreibung der Umsetzung der Anforderung erfolgt. Das Aussehen einer Anforderungstabelle zu der beispielhaft erstellten und verworfenen Anforderung Chat-1T1 wäre wie folgt:

ID	`Chat-1T1`
Name	Übermittlung
Verbindlichkeit	muss
Status	verworfen (s. S. 38)
Erstellt am	23.06.2015
Ziel	Eine Nachricht soll an einen anderen Nutzer übermittelt werden
Kurz-beschreibung	Hier können mehrere Zeilen Text stehen. Der Text wird im Blockformat angezeigt und automatisch umgebrochen.
Fehlerfall	Person A schreibt eine Text-Nachricht an Person B und schickt diese ab. Die Nachricht kommt nicht bei Nutzer B an.
Akteure	- Person A (schreibt) - Person B (empfängt) - Server
Vorbedingung	Person A ist angemeldet.
Nachbedingung	Person B hat die Text-Nachricht erhalten.
Fachlicher Auslöser	Person A bestätigt ihre Eingabe
Ausgetauschte Information	- Nachricht zwischen A und B
Anmerkungen	Dies ist eine Beispiel-Anforderung

Im Folgenden werden die jeweiligen HLUC beschrieben und durch ihre Aktivitätsdiagramm veranschaulicht. Dazu werden die einzelnen PUC des jeweiligen HLUC durch ihre Anforderungstabellen aufgezeigt. In diesen sind des Weiteren Referenzen zu den Aktivitätsdiagrammen des zu betrachtenden PUC und seinen Akteuren gegeben, welche sich im Anhang der Dokumentation befinden.

High Level Use Case *Virtuelles Kraftwerk initialisieren* Bei der Initialisierung des virtuellen Kraftwerks wird zunächst dessen Controller gestartet, welcher die Konfigurationsdateien lädt. Anschließend werden die Programmmodule gestartet und ein internes Kommunikationsnetzwerk aufgebaut. Dieses umfasst insbesondere den Zugriff auf die interne Datenbank, welche ebenfalls initialisiert wird. Zusätzlich werden Schnittstellen zum Markt- und Wetterdatenanbieter, ebenso wie zu den Anlagen (über die Simulationsumgebung) bereitgestellt. In Abbildung 7 das Aktivitätsdiagramm für diesen HLUC angegeben. In dieser sind die einzelnen PUCs zu sehen, die zusammen den HLUC *Virtuelles Kraftwerk initialisieren* bilden, sowie der Ablauf der auszuführenden Tätigkeiten der PUCs. Eine Übersicht aller PUCs diese HLUCs kann dem Anhang D.1.1 entnommen werden.

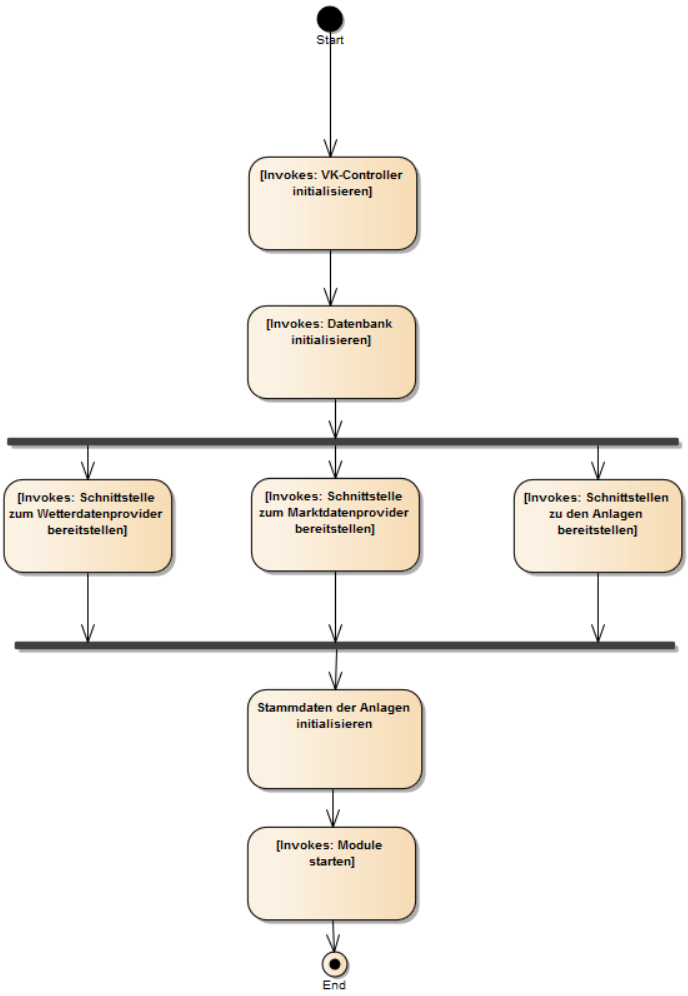


Abbildung 7: HLUC *Virtuelles Kraftwerk initialisieren* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Virtuelles Kraftwerk initialisieren*

ID	VPP-PUC1
Name	VK-Controller initialisieren
Verbindlichkeit	muss
Status	umgesetzt (s. S. 110)
Erstellt am	31.08.2015
Ziel	Der VK-Controller ist initial gestartet.
Kurz-beschreibung	Der VK-Admin startet den VK-Controller mit den benötigten Startparametern.
Fehlerfall	- Die Parameter sind fehlerhaft gesetzt. - Die Parameterformatierung ist falsch gewählt. - Die Ausführungsumgebung des Systems ist nicht installiert. - Der VK-Admin besitzt keine ausreichenden Ausführungsrechte.
Akteure	- VK-Admin - VK-Controller (s. Akteurabbildung: 69)
Vorbedingung	- Der VK-Admin besitzt Ausführungsrechte für das lauffähige System. - Der VK-Admin hat die Startparameter bereitgestellt.
Fachlicher Auslöser	Die Dienste des Systems werden benötigt.
Ausgetauschte Information	- Es werden keine Informationen ausgetauscht (s. Aktivitätsdiagramm: 69)

ID	VPP-PUC2
Name	Module starten
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	31.08.2015
Ziel	Der Produktportfoliooptimierer, der Einsatzplanoptimierer, das Wettermodul, das Marktmodul und das Flexibilitätenmodul sind gestartet.
Kurzbeschreibung	Der VK-Controller startet den Produktportfoliooptimierer, den Einsatzplanoptimierer, das Wettermodul, das Marktmodul und das Flexibilitätenmodul.
Fehlerfall	- Zu einem oder mehreren Modulen konnte keine Verbindung aufgebaut werden. - Eines oder mehrere der Module können nicht gestartet werden.
Akteure	- VK-Controller - Produktportfoliooptimierer - Einsatzplanoptimierer - Wettermodul - Marktmodul - Flexibilitätenmodul (s. Akteurabbildung: 70)
Vorbedingung	VPP-PUC1, VPP-PUC3
Fachlicher Auslöser	Die Module werden für das Ausführen der einzelnen Aufgaben benötigt.
Ausgetauschte Information	- Stammdatenblätter der Anlagen - Stammdaten der Anlagen (s. Aktivitätsdiagramm: 70)

ID	VPP-PUC3
Name	Lokale Datenbank initialisieren
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	31.08.2015
Ziel	Der VK-Controller muss über eine initialisierte Instanz der lokalen Datenbank verfügen.
Kurzbeschreibung	Der VK-Controller erstellt mit Hilfe der benötigten Verbindungsdaten eine Instanz der lokalen Datenbank.
Fehlerfall	- Die Verbindung schlägt fehl.
Akteure	- VK-Controller - lokale Datenbank (s. Akteurabbildung: 71)
Vorbedingung	VPP-PUC1
Fachlicher Auslöser	Nach dem Start des Systems sollen die Daten aktualisiert werden. Der Zugriff auf die Daten soll vorbereitet werden.
Ausgetauschte Information	- Datenbankverbindung (s. Aktivitätsdiagramm: 71)

ID	VPP-PUC4
Name	Schnittstelle zum Wetterdatenprovider bereitstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	26.06.2015
Ziel	Der VK-Controller muss über eine Schnittstelle zum Wetterdatenprovider verfügen.
Kurzbeschreibung	Der VK-Controller baut eine Verbindung zum Wetterdatenprovider auf und stellt die Schnittstelle systemintern bereit.
Fehlerfall	- Es kann keine Verbindung zum Wetterdatenprovider hergestellt werden.
Akteure	- VK-Controller - Wetterdatenprovider (s. Akteurabbildung: 72)
Vorbedingung	VPP-PUC1
Fachlicher Auslöser	Die bereits vorliegenden Wetterdaten in der lokalen Datenbank sollen aktualisiert werden.
Ausgetauschte Information	- Wetterdatenbankverbindung (s. Aktivitätsdiagramm: 72)

ID	VPP-PUC5
Name	Schnittstelle zum Marktdatenprovider bereitstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	26.06.2015
Ziel	Der VK-Controller muss über eine Schnittstelle zum Marktdatenprovider verfügen.
Kurz-beschreibung	Der VK-Controller baut eine Verbindung zum Marktdatenprovider auf und stellt die Schnittstelle systemintern bereit.
Fehlerfall	- Es kann keine Verbindung zum Marktdatenprovider hergestellt werden.
Akteure	- VK-Controller - Marktdatenprovider (s. Akteurabbildung: 73)
Vorbedingung	VPP-PUC1
Fachlicher Auslöser	Die bereits vorliegenden Marktdaten in der lokalen Datenbank sollen aktualisiert werden.
Ausgetauschte Information	- Marktdatenbankverbindung (s. Aktivitätsdiagramm: 73)

ID	VPP-PUC7
Name	Schnittstelle zu den Anlagen bereitstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 112)
Erstellt am	26.06.2015
Ziel	Der VK-Controller muss über eine Schnittstelle zu den Anlagen verfügen
Kurz-beschreibung	Der VK-Controller baut eine Verbindung zu den Anlagen auf und stellt die Schnittstelle systemintern bereit.
Fehlerfall	- Es kann keine Verbindung zu einer oder mehreren Anlagen hergestellt werden.
Akteure	- VK-Controller - Anlage (s. Akteurabbildung: 74)
Vorbedingung	VPP-PUC1
Fachlicher Auslöser	Das System soll in der Lage sein, den Anlagen einen Fahrplan zu übermitteln und die Zustände der Anlagen abzufragen.
Ausgetauschte Information	- Anlagenverbindung (s. Aktivitätsdiagramm: 74)

ID	VPP-PUC8
Name	Stammdaten der Anlagen initialisieren
Verbindlichkeit	muss
Status	umgesetzt (s. S. 112)
Erstellt am	15.09.2015
Ziel	Die Stammdaten der Anlagen müssen im VK-Controller in einer Form vorliegen, die auch von den anderen Modulen genutzt werden kann.
Kurzbeschreibung	Der VK-Controller greift auf die Stammdatenblätter (txt-Dateintyp) zu, die vom Anlagenbetreiber bereitgestellt wurden, liest die Stammdaten aus, konvertiert sie in eine verarbeitbare Form und speichert sie lokal ab, sodass diese auch von anderen Modulen genutzt werden können.
Fehlerfall	- Das VK-Controller kann die Stammdatenblätter nicht einlesen.
Akteure	- VK-Controller - Anlagenbetreiber - lokale Datenbank (s. Akteurabbildung: 75)
Vorbedingung	Der VK-Controller ist hochgefahren. Die Anlagenbetreiber haben Stammdatenblätter in dem vom System vorgeschriebenen Format zur Verfügung gestellt.
Fachlicher Auslöser	Das VK-Controller benötigt die Anlagenstammdaten zur Registrierung der Anlagen und zur Berechnung der Flexibilitäten. Die Anlagenstammdaten werden vom VK-Controller zur Registrierung der Anlagen und zur Berechnung der Flexibilitäten benötigt.
Ausgetauschte Information	- Stammdatenblätter der Anlagen - Stammdaten der Anlagen (s. Aktivitätsdiagramm: 75)

High Level Use Case *Wetterprognose erstellen* Das Wettermodul fragt beim Wetterdatenanbieter einmalig historische Wetterprognosen und historische Wetterdaten ab und speichert diese in der lokalen Datenbank. Dies ist ausreichend, da die Funktion des Produktes auf einen historischen Betrieb beschränkt wurde. Während diesem wird wiederholend eine der historischen Wetterprognosen aus der lokalen Datenbank geholt, um für den entsprechenden Tag eine Day-Ahead-Planung durchführen zu können. Anhand der Wetterprognose wird ein Datenfilter erstellt, um aus der Wetterdatenbank die Tage/Zeiträume herauszusuchen, an denen ein ähnliches Wetter herrschte, wie am Wetterprognosetag. Die Datenbankanfrageergebnisse werden an das Marktmodul gesendet. Eine Übersicht kann dem Anhang D.1.2 und das Aktivitätsdiagramm der Abbildung 8 entnommen werden.

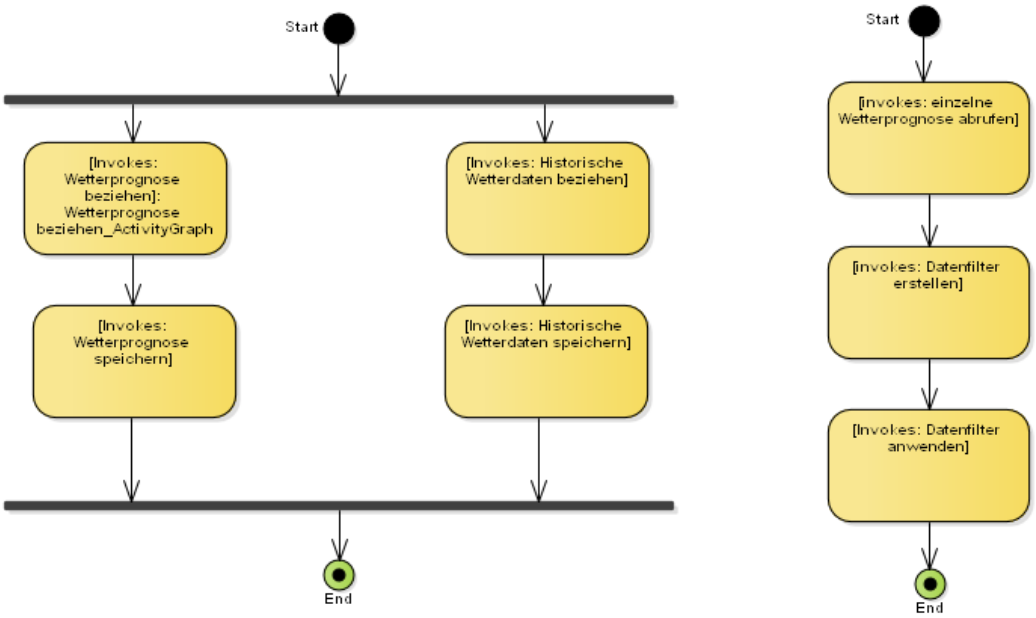


Abbildung 8: HLUC *Wettervorhersage* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Wetterprognose erstellen*

ID	`Wetter-PUC1`
Name	Historische Wetterdaten beziehen
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	25.06.2015
Ziel	Die seit der letzten Aktualisierung der lokalen Datenbank neu beim Wetterdatenprovider hinzugekommenen Wetterdaten stehen dem Wettermodul zur Verfügung.
Kurzbeschreibung	Das Wettermodul des Systems sendet die Datenbankanfrage an den Wetterdatenprovider. Der Provider liefert die angeforderten Daten.
Fehlerfall	- Das Format der Wetterdaten ist verändert, sodass diese vom Wettermodul nicht mehr ausgelesen werden können. - Der Wetterdatenprovider verweigert die Herausgabe der Daten.
Akteure	- Wettermodul - Wetterdatenprovider (s. Akteurabbildung: 75)
Vorbedingung	Datenbankanfrage steht zur Verfügung und die Verbindung zum Wetterdatenprovider ist aufgebaut (VPP-PUC4)
Fachlicher Auslöser	Das Wettermodul benötigt möglichst aktuelle Wetterdaten.
Ausgetauschte Information	- Wetterdaten (s. Aktivitätsdiagramm: 77)

ID	`Wetter-PUC2`
Name	Historische Wetterdaten speichern
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	30.08.2015
Ziel	Die Wetterdaten sind in der lokalen Datenbank gespeichert.
Kurz-beschreibung	Die Wetterdaten werden vom Wettermodul mit Hilfe einer Datenbankanfrage in die lokale Datenbank gespeichert.
Fehlerfall	- Die Wetterdaten können nicht erfolgreich in die lokale Datenbank gespeichert werden
Akteure	- Wettermodul - lokale Datenbank (s. Akteurabbildung: 78)
Vorbedingung	Wetter-PUC1
Fachlicher Auslöser	Das Wettermodul benötigt eine möglichst umfassende aktuelle Wetterdatensammlung.
Ausgetauschte Information	- Wetterdaten (s. Aktivitätsdiagramm: 78)

ID	`Wetter-PUC3`
Name	Wetterprognose beziehen
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	25.06.2015
Ziel	Die Wetterprognose für den betrachteten Tag steht dem Wettermodul zur Verfügung.
Kurz-beschreibung	Das Wettermodul ruft die Wetterprognose für den zu betrachtenden Tag mit Hilfe einer Datenbankanfrage vom Wetterdatenprovider ab.
Fehlerfall	- Für den betrachteten Tag ist keine Wetterprognose vorhanden. - Das Format der Wetterdaten wurde verändert, sodass diese vom Wettermodul nicht mehr ausgelesen werden können. - Der Wetterdatenprovider verweigert die Herausgabe der Daten.
Akteure	- Wettermodul - Wetterdatenprovider (s. Akteurabbildung: 93)
Vorbedingung	Wetter-PUC2, VPP-PUC4
Fachlicher Auslöser	Zur Berechnung der Tage, die ein ähnliches Wetter wie der betrachtete Tag haben, wird dessen Wetterprognose benötigt.
Ausgetauschte Information	- Login-Daten - Wetterprognose (s. Aktivitätsdiagramm: 93)
Anmerkungen	Im Unterschied zu Wetter-PUC5 werden in diesem PUC die Wetterdaten vom Wetterdatenprovider erstmalig angefordert.

ID	`Wetter-PUC4`
Name	Wetterprognose speichern
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	25.06.2015
Ziel	Die Wetterprognose ist in der lokalen Datenbank gespeichert.
Kurz-beschreibung	Die Wetterprognose wird vom Wettermodul mit Hilfe einer Datenbankanfrage in die lokale Datenbank gespeichert.
Fehlerfall	- Die Wetterprognose kann nicht erfolgreich in die lokale Datenbank gespeichert werden.
Akteure	- Wettermodul - lokale Datenbank (s. Akteurabbildung: 80)
Vorbedingung	Wetter-PUC3
Fachlicher Auslöser	Die Wetterprognose soll gespeichert werden, damit in der weiteren Optimierung auf diese zugegriffen werden kann.
Ausgetauschte Information	- Wetterprognose (s. Aktivitätsdiagramm: 80)

ID	`Wetter-PUC5`
Name	Einzelne Wetterprognose abrufen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 118)
Erstellt am	25.06.2015
Ziel	Die Wetterprognose für den betrachteten Tag steht dem Wettermodul zur Verfügung.
Kurz-beschreibung	Das Wettermodul ruft die Wetterprognose für den zu betrachtenden Tag mit Hilfe einer Datenbankanfrage aus der lokalen Datenbank ab.
Fehlerfall	- Für den betrachteten Tag ist keine Wetterprognose vorhanden.
Akteure	- Wettermodul - lokale Datenbank (s. Akteurabbildung: 81)
Vorbedingung	Wetter-PUC2
Fachlicher Auslöser	Zur Berechnung der Tage, die ein ähnliches Wetter wie der betrachtete Tag haben, wird dessen Wetterprognose benötigt.
Ausgetauschte Information	- Wetterprognose (s. Aktivitätsdiagramm: 81)
Anmerkungen	Im Unterschied zu Wetter-PUC3 werden in diesem PUC Wetterdaten täglich aus der lokalen Datenbank bezogen.

ID	`Wetter-PUC6`
Name	Datenfilter erstellen
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	25.06.2015
Ziel	Das Wettermodul muss eine Datenbankanfrage auf Basis der Wetterprognose erstellen.
Kurzbeschreibung	Das Wettermodul nimmt die Charakteristiken der Wetterprognose um einen Datenfilter für die Datenbankanfrage zu erstellen.
Fehlerfall	- Die Datenbankanfrage wird fehlerhaft erstellt.
Akteure	- Wettermodul (s. Akteurabbildung: 82)
Vorbedingung	Wetter-PUC5
Fachlicher Auslöser	Wettermodul benötigt eine Datenbankanfrage, um ähnliche Tage zum betrachteten Tag zu finden.
Ausgetauschte Information	- Es werden keine Informationen ausgetauscht (s. Aktivitätsdiagramm: 82)

ID	`Wetter-PUC7`
Name	Datenfilter anwenden
Verbindlichkeit	muss
Status	verworfen (s. S. 118)
Erstellt am	1.07.2015
Ziel	Die ähnlichen Tage sind aus der Datenbank ausgelesen.
Kurzbeschreibung	Der erstellte Datenfilter wird auf die historischen Wetterdaten in der lokalen Datenbank angewendet.
Fehlerfall	- Die Daten können nicht ausgelesen werden.
Akteure	- Wettermodul - lokale Datenbank (s. Akteurabbildung: 83)
Vorbedingung	Wetter-PUC6
Fachlicher Auslöser	Das Marktmodul benötigt die gefilterten historischen Wetterdaten zur Erstellung der Marktprognose.
Ausgetauschte Information	- Wetterdaten (s. Aktivitätsdiagramm: 83)

High Level Use Case *Marktprognose erstellen* Das Marktmodul erstellt aus historischen Markt- und Wetterdaten eine Marktprognose. Dazu werden historische Wettervorhersagen und -daten, sowie die aktuelle Wettervorhersage genutzt. Um relevante Marktdaten zu erhalten, wird ein angepasster Marktdatenfilter erstellt. Dabei werden geeignete Attribute ausgewählt und sinnvolle Toleranzbereiche für diese festgelegt. Dieser Filter wird als Datenbankanfrage formuliert und ausgeführt. Mit Hilfe

der erhaltenen Daten wird eine Marktprognose erstellt. Die Marktprognose wird anschließend an den Produktportfoliooptimierer übergeben. Damit die lokale Datenbank mit historischen Marktdaten aktuell bleibt, werden neue Marktdaten gelegentlich beim Marktdatenanbieter angefragt. Die Übersicht befindet sich analog zu den vorherigen HLUCs im Anhang D.1.3 und das Aktivitätsdiagramm kann der folgenden Abbildung 9 entnommen werden.

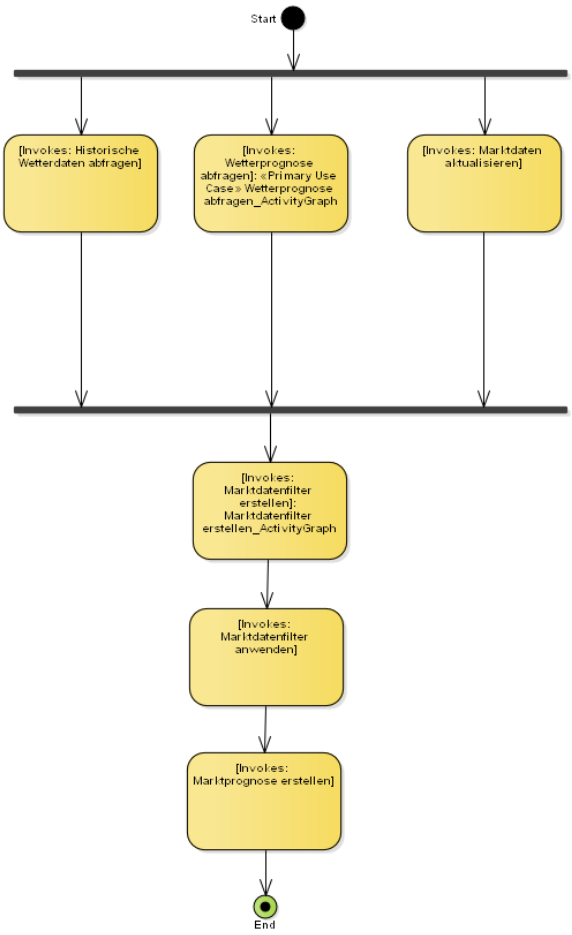


Abbildung 9: HLUC *Marktprognose* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Marktprognose erstellen*

ID	`Markt-PUC1`
Name	Marktdaten aktualisieren
Verbindlichkeit	muss
Status	verworfen (s. S. 126)
Erstellt am	15.09.2015
Ziel	Die Marktdaten in der lokalen Datenbank sind um Marktdaten ergänzt, die aktueller als die bisher gespeicherten sind.
Kurz-beschreibung	Das Marktmodul sendet eine Anfrage an den Marktdatenprovider, um aktuellere Marktdaten zu erhalten, die sich noch nicht in der lokalen Datenbank befinden. Das Marktmodul erhält die Marktdaten, konvertiert diese in ein geeignetes Format und speichert sie in die lokale Datenbank.
Fehlerfall	- Die Marktdaten werden nicht erfolgreich übermittelt. - Der Marktdatenprovider verweigert die Herausgabe der Daten. - Das Format der Marktdaten wurde verändert, sodass diese vom Marktmodul nicht mehr ausgelesen werden können. - Die Marktdaten können nicht erfolgreich in die lokale Datenbank gespeichert werden.
Akteure	- Marktmodul - Marktdatenprovider - lokale Datenbank (s. Akteurabbildung: 85)
Vorbedingung	Verbindung zum Marktdatenprovider (VPP-PUC4)
Fachlicher Auslöser	Das Marktmodul möchte zur Erstellung der Marktprognose aktualisierte Marktdaten verwenden.
Ausgetauschte Information	- Marktpreise (s. Aktivitätsdiagramm: 85)

ID	`Markt-PUC2`
Name	Wetterprognose abfragen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	25.06.2015
Ziel	Die Wetterprognose liegt im Marktmodul vor.
Kurz-beschreibung	Das Marktmodul sendet eine Anfrage an das Wettermodul bezüglich der Wetterprognose. Das Marktmodul erhält die Wetterprognose. Das Marktmodul konvertiert die Wetterprognose ggf. in ein geeignetes Format.
Fehlerfall	- Die Wetterprognose wird nicht erfolgreich übermittelt.
Akteure	- Wettermodul - Marktmodul (s. Akteurabbildung: 86)
Vorbedingung	Verbindung zum Wettermodul
Fachlicher Auslöser	Das Marktmodul benötigt die Wetterprognose zur Erstellung der Marktprognose.
Ausgetauschte Information	- Wetterprognose (s. Aktivitätsdiagramm: 86)

ID	`Markt-PUC3`
Name	Historische Wetterdaten abfragen
Verbindlichkeit	muss
Status	Umgesetzt (s. S. 122)
Erstellt am	25.06.2015
Ziel	Die historischen Wetterdaten liegen geeignet formatiert im Marktmodul vor.
Kurz-beschreibung	Das Marktmodul sendet eine Anfrage an das Wettermodul, um die nach ähnlichen Tagen gefilterten historischen Wetterdaten zu erhalten. Das Marktmodul erhält diese und konvertiert sie ggf. in ein geeignetes Format.
Fehlerfall	- Die Wetterdaten werden nicht erfolgreich übermittelt. - Die falschen Wetterdaten werden übermittelt. - Das Format der Wetterdaten wurde verändert, sodass diese vom Marktmodul nicht mehr ausgelesen werden können.
Akteure	- Wettermodul - Marktmodul (s. Akteurabbildung: 87)
Vorbedingung	Verbindung zum Wettermodul
Fachlicher Auslöser	Das Marktmodul benötigt die Wetterdaten zur Erstellung der Marktprognose.
Ausgetauschte Information	- Wetterdaten (s. Aktivitätsdiagramm: 87)

ID	`Markt-PUC4`
Name	Marktdatenfilter erstellen
Verbindlichkeit	muss
Status	verworfen (s. S. 126)
Erstellt am	30.08.2015
Ziel	Das Marktmodul verfügt über einen Marktdatenfilter.
Kurz-beschreibung	Das Marktmodul erstellt einen Filter, der für die vom Wettermodul herausgefundenen Tage mit ähnlichem Wetter die Marktdaten anfragen kann.
Fehlerfall	- Es wurde kein einsatzbereiter Marktdatenfilter erstellt. - Es wurde ein falscher Marktdatenfilter erstellt.
Akteure	- Marktmodul (s. Akteurabbildung: 87)
Vorbedingung	Markt-PUC1, Markt-PUC2, Markt-PUC3
Fachlicher Auslöser	Das Marktmodul benötigt einen Marktdatenfilter, um die Marktpreise der relevanten Tage aus der lokalen Datenbank auszulesen.
Ausgetauschte Information	- Es werden keine Informationen ausgetauscht (s. Aktivitätsdiagramm: 88)

ID	`Markt-PUC5`
Name	Marktdatenfilter anwenden
Verbindlichkeit	muss
Status	verworfen (s. S. 126)
Erstellt am	25.06.2015
Ziel	Marktmodul verfügt über die gefilterten historischen Marktdaten.
Kurz-beschreibung	Der Marktdatenfilter wird angewendet und die gefilterten Daten werden aus der lokalen Datenbank ausgelesen.
Fehlerfall	- Es werden keine gefilterten Marktdaten zurückgeliefert.
Akteure	- Marktmodul - lokale Datenbank (s. Akteurabbildung: 89)
Vorbedingung	Markt-PUC4
Fachlicher Auslöser	Das Marktmodul benötigt gefilterte Marktdaten, um aus diesen die Marktprognose des gewünschten Tages zu erstellen.
Ausgetauschte Information	- Marktpreise (s. Aktivitätsdiagramm: 89)

ID	`Markt-PUC6`
Name	Marktprognose erstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 122)
Erstellt am	25.06.2015
Ziel	Die Marktprognose des gewünschten Tages liegt vor.
Kurz-beschreibung	Das Marktmodul erstellt anhand der historischen Marktdaten eine Marktprognose für den gewünschten Tag.
Fehlerfall	- Es wird keine Marktprognose erstellt.
Akteure	- Marktmodul (s. Akteurabbildung: 90)
Vorbedingung	Markt-PUC5
Fachlicher Auslöser	Das Marktmodul benötigt eine Marktprognose
Ausgetauschte Information	- Marktpreise - Marktprognose (s. Aktivitätsdiagramm: 90)

High Level Use Case *Flexibilitäten ermitteln* Mithilfe einer Simulationsumgebung werden Anlagen simuliert, um gültige Flexibilitäten zu erhalten. Flexibilitäten umfassen den Rahmen möglicher Fahrpläne einzelner Anlagen eines Tages. Mit Hilfe der Flexibilitäten sollen später gültige Fahrpläne erstellt werden, die das vorher festgelegte Produktportfolio erfüllen. Das Aktivitätsdiagramm zum vorliegenden HLUC ist in der Abbildung 10 dargestellt. Eine Übersicht aller PUCs befindet sich im Anhang D.1.4.

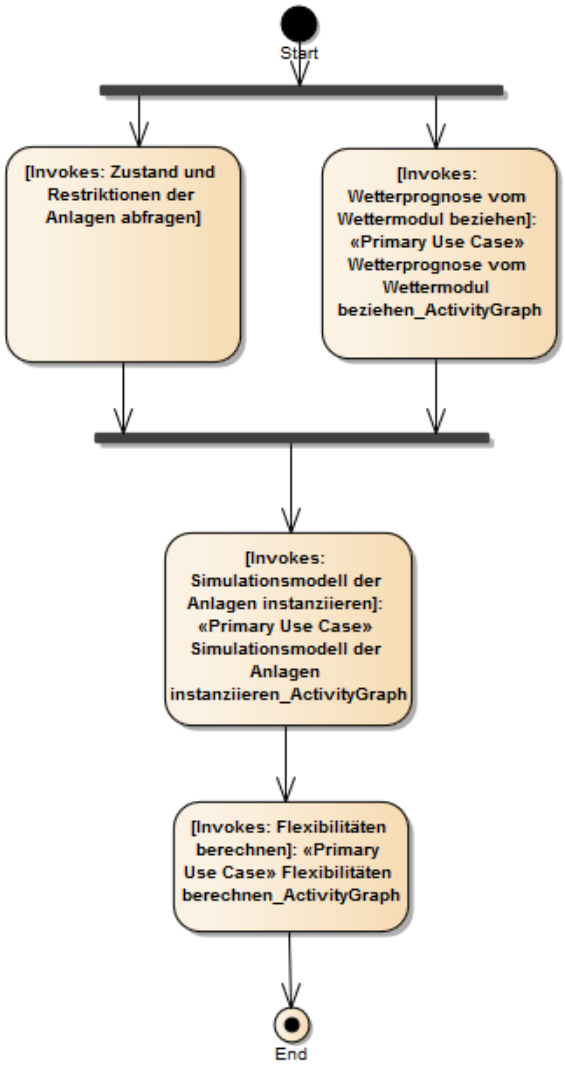


Abbildung 10: HLUC *Flexibilitäten ermitteln* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Flexibilitäten ermitteln*

ID	`Flex-PUC1`
Name	Stammdaten mit Zustand der Anlage abfragen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 105)
Erstellt am	26.06.2015
Ziel	Zustände und Stammdaten der Anlagen liegen im Flexibilitätenmodul vor.
Kurz-beschreibung	Das Flexibilitätenmodul sendet Anfragen an die einzelnen, zuvor ermittelten, registrierten Anlagen bezüglich ihres Zustands und ihrer Stammdaten. Die Anlagen senden den Zustand und die Stammdaten zurück.
Fehlerfall	- Das Flexibilitätenmodul sendet die Anfragen und erhält keine Rückmeldung. - Falsche Interpretation der Anfrage durch die Anlagen führt zu fehlerhaften Rückmeldungen. - Rückmeldung der Anlage im falschen Format.
Akteure	- Flexibilitätenmodul - Anlagen - lokale Datenbank (s. Akteurabbildung: 92)
Vorbedingung	VPP-PUC7
Fachlicher Auslöser	Das Flexibilitätenmodul benötigt die Zustände der Anlagen, um daraus die Flexibilitäten der Anlagen zu berechnen.
Ausgetauschte Information	- Zustände der Anlagen - Stammdaten der Anlagen (s. Aktivitätsdiagramm: 92)

ID	`Flex-PUC2`
Name	Wetterprognose vom Wettermodul beziehen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 111)
Erstellt am	26.06.2015
Ziel	Die Wetterprognose für den gewünschten Tag liegt im Flexibilitätenmodul vor.
Kurz-beschreibung	Das Flexibilitätenmodul sendet eine Anfrage an das Wettermodul bezüglich der Wetterprognose für den gewünschten Tag. Das Flexibilitätenmodul erhält diese Wetterprognose.
Fehlerfall	- Die Wetterprognose wird nicht erfolgreich übermittelt.
Akteure	- Wettermodul - Flexibilitätenmodul (s. Akteurabbildung: 93)
Vorbedingung	Verbindung zum Wettermodul
Fachlicher Auslöser	Das Flexibilitätenmodul benötigt die Wetterprognose, um die Anlagen wetterabhängig zu simulieren.
Ausgetauschte Information	- Wetterdaten (s. Aktivitätsdiagramm: 93)

ID	`Flex-PUC3`
Name	Simulationsmodell der Anlagen instanziieren
Verbindlichkeit	muss
Status	umgesetzt (s. S. 127)
Erstellt am	26.06.2015
Ziel	Im Flexibilitätenmodul ist eine Instanz des Simulationsmodells der Anlagen vorhanden.
Kurz-beschreibung	Das Flexibilitätenmodul instanziiert für jede Anlage ein Modell mit den entsprechenden Parametern, die sich aus den Zuständen und den Stammdaten der Anlagen ergeben. Das Flexibilitätenmodul erstellt eine Instanz der Simulationsumgebung, welche die Modelle der einzelnen Anlagen beinhaltet.
Fehlerfall	- Das Simulationsmodell mindestens einer Anlage wird nicht erfolgreich bzw. fehlerhaft erstellt.
Akteure	- Flexibilitätenmodul (s. Akteurabbildung: 93)
Vorbedingung	Wetterprognose sowie Zustände und Stammdaten der Anlagen liegen vor (Flex-PUC1, Flex-PUC2)
Fachlicher Auslöser	Das Flexibilitätenmodul muss die beim System registrierten Anlagen simulieren, um deren Flexibilitäten berechnen zu können.
Ausgetauschte Information	- Zustände der Anlagen - Stammdaten der Anlagen (s. Aktivitätsdiagramm: 94)

ID	`Flex-PUC4`
Name	Flexibilitäten berechnen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 127)
Erstellt am	26.06.2015
Ziel	Das Flexibilitätenmodul verfügt über die Flexibilitäten der Anlagen.
Kurz-beschreibung	Das Flexibilitätenmodul führt die Modelle innerhalb der Simulationsumgebung aus und erhält dadurch die Flexibilitäten zurück.
Fehlerfall	- Die Flexibilitäten werden nicht berechnet. - Die Simulation weicht von der Realität ab (Modellfehler).
Akteure	- Flexibilitätenmodul (s. Akteurabbildung: 95)
Vorbedingung	Flex-PUC3
Fachlicher Auslöser	Der Einsatzplanoptimierer benötigt die Flexibilitäten. Der Produktportfoliooptimierer benötigt die Flexibilitäten.
Ausgetauschte Information	- Wetterdaten - Stammdaten der Anlagen - Flexibilitäten (s. Aktivitätsdiagramm: 95)

High Level Use Case *Einsatzplanoptimierung* Der Einsatzplanoptimierer wird zu zwei Zwecken aufgerufen. Einmalig, um initial ein Referenzanlagenportfolio und daraus eine Referenzleistungskurve für die Produktportfoliooptimierung zu erstellen, und später während jedem Optimierungsschritt des Produktportfoliooptimierers, um einen optimierten Einsatzplan zu erhalten. Der Ablauf dieses HLUCs ist aus dem Aktivitätsdiagramm 11 entnehmbar.

Für die Erstellung des initialen Referenzanlagenportfolios werden die Flexibilitäten ermittelt. Mithilfe derselben wird das Referenzanlagenportfolio erstellt, das beispielsweise nur die PV-Anlagen enthält. Die sich daraus ergebende Referenzleistungskurve wird vom Produktportfoliooptimierer verwendet, um einen gültigen Startpunkt für die Optimierung zu bestimmen. Die Referenzleistungskurve wird an den Produktportfoliooptimierer weitergegeben. Während dieser das Produktportfolio optimiert, wird für eine sich daraus ergebende Lastkurve und die bestehenden Flexibilitäten ein optimaler Einsatzplan benötigt. Dieser wird vom Einsatzplanoptimierer erstellt. Dies wird in jedem Schritt wiederholt. Eine Übersicht befinden sich im Anhang D.1.5.

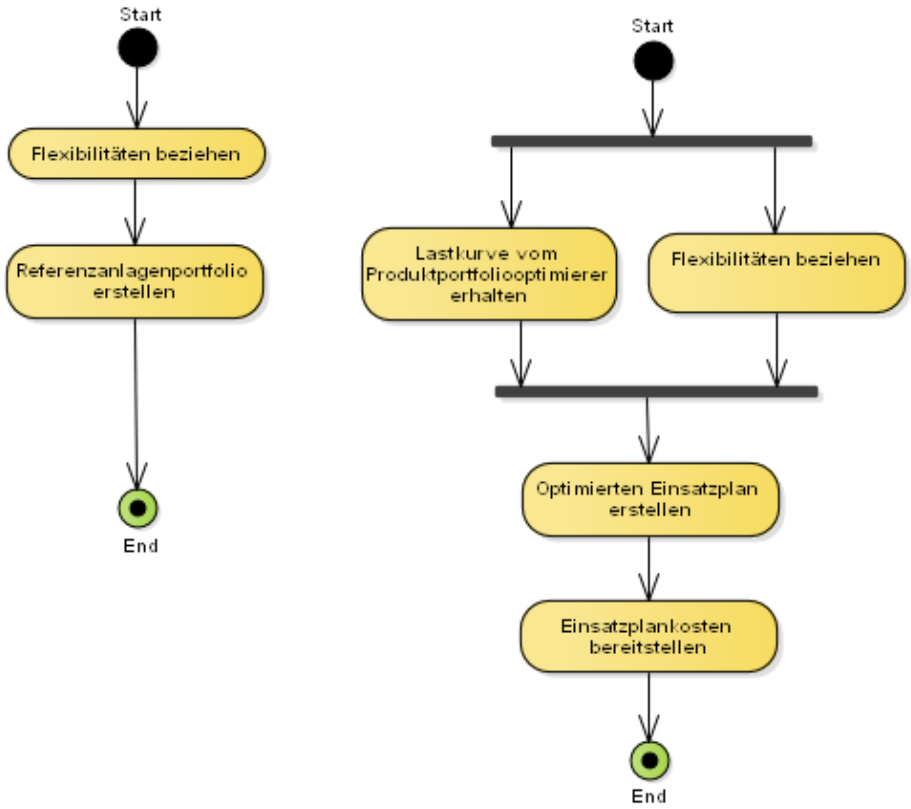


Abbildung 11: HLUC *Einsatzplanoptimierung* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Einsatzplanoptimierung*

ID	EPO-PUC1
Name	Flexibilitäten beziehen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Der Einsatzplanoptimierer verfügt über die Flexibilitäten der einzelnen Anlagen.
Kurzbeschreibung	Der Einsatzplanoptimierer fragt die Flexibilitäten vom Flexibilitätenmodul ab.
Fehlerfall	- Die Flexibilitäten werden nicht bzw. fehlerhaft übertragen.
Akteure	- Flexibilitätenmodul - Einsatzplanoptimierer (s. Akteurabbildung: 97)
Vorbedingung	Verbindung zum Flexibilitätenmodul.
Fachlicher Auslöser	Einsatzplanoptimierer benötigt die Flexibilitäten, um einen Einsatzplan zu erstellen und zu optimieren.
Ausgetauschte Information	- Flexibilitäten (s. Aktivitätsdiagramm: 97)

ID	EPO-PUC2
Name	Referenzanlagenportfolio erstellen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Das Referenzanlagenportfolio und die zugehörige Referenzleistungskurve liegt vor.
Kurz-beschreibung	Der Einsatzplanoptimierer erstellt ein Referenzanlagenportfolio, das beispielsweise nur die Anlagen mit den geringsten Betriebskosten beinhaltet. Anschließend berechnet er die sich ergebende Referenzleistungskurve.
Fehlerfall	- Das Referenzanlagenportfolio oder die Referenzleistungskurve werden nicht erstellt.
Akteure	Einsatzplanoptimierer (s. Akteurabbildung: 98)
Vorbedingung	EPO-PUC1
Fachlicher Auslöser	Der Einsatzplanoptimierer benötigt das Referenzanlagenportfolio für den ersten Optimierungsschritt.
Ausgetauschte Information	- Flexibilitäten - Lastkurve - Referenzanlagenportfolio -Referenzleistungskurve mit Kosten (s. Aktivitätsdiagramm: 98)

ID	EPO-PUC3
Name	Lastkurve vom Portfoliooptimierer erhalten
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	26.06.2015
Ziel	Der Einsatzplanoptimierer verfügt über die Lastkurve der Produktportfolio-optimierung.
Kurz-beschreibung	Der Einsatzplanoptimierer erhält die Lastkurve vom Portfoliooptimierer.
Fehlerfall	- Die Lastkurve wird nicht übertragen.
Akteure	- Portfoliooptimierer - Einsatzplanoptimierer (s. Akteurabbildung: 99)
Vorbedingung	Verbindung zum Portfoliooptimierer
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt den optimierten Einsatzplan für seinen weiteren Optimierungsschritt.
Ausgetauschte Information	- Produktportfolio - Lastkurve (s. Aktivitätsdiagramm: 99)

ID	EPO-PUC4
Name	Optimierten Einsatzplan erstellen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Der Einsatzplanoptimierer erstellt einen optimierten Einsatzplan.
Kurz-beschreibung	Der Einsatzplanoptimierer erstellt zu der vom Produktportfoliooptimierer erhaltenen Lastkurve einen optimierten Einsatzplan und berechnet die Kosten für diesen.
Fehlerfall	- Der Einsatzplan wird nicht (optimal) erstellt.
Akteure	- Einsatzplanoptimierer (s. Akteurabbildung: 100)
Vorbedingung	EPO-PUC3
Fachlicher Auslöser	Für eine Lastkurve muss ein optimaler Einsatzplan erstellt werden, damit der Produktportfoliooptimierer diesen in der weiteren Optimierung verwenden kann.
Ausgetauschte Information	- Flexibilitäten - Lastkurve - Einsatzplan (s. Aktivitätsdiagramm: 100)

ID	EPO-PUC5
Name	Einsatzplankosten zurückgeben
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Der Produktportfoliooptimierer verfügt über die Kosten des Einsatzplans.
Kurz-beschreibung	Der Einsatzplanoptimierer gibt die Kosten des Einsatzplans an den Produkt-portfoliooptimierer.
Fehlerfall	- Kosten werden nicht bereitgestellt.
Akteure	- Einsatzplanoptimierer - Produktportfoliooptimierer (s. Akteurabbildung: 101)
Vorbedingung	EPO-PUC4
Fachlicher Auslöser	Der Portfoliooptimierer benötigt die Einsatzplankosten.
Ausgetauschte Information	- Einsatzplankosten (s. Aktivitätsdiagramm: 101)

High Level Use Case *Produktportfoliooptierung* Der Produktportfoliooptimierer erstellt ein Produktportfolio, welches darauf optimiert wurde, den größtmöglichen Gewinn zu erzielen. Dazu wird zunächst beim Marktmodul die aktuelle Marktprognose abgefragt. Gleichzeitig wird vom Einsatzplanoptimierer eine Referenzeinsatzplan bezogen, welcher auf den einzelnen Fahplänen auf Basis von Leistungskurven aufbaut. Da der Referenzeinsatzplan auf den addierten Leistungskurven besteht, kann dieser auch als Referenzleistungskurve bezeichnet werden. Mit dem Referenzeinsatzplan werden beim Optimierungsvorgang verschiedene optimierte Produktportfolios berechnet. Die Erstellung eines einzelnen optimierten Produkportfolios sieht dabei so aus, dass zunächst anhand der Referenzeinsatzplans Produkte der Strombörse ausgewählt werden. Anschließend wird die Marktprognose dazu verwendet, Preise für diese Produkte festzulegen. Aus den Produkten und deren Preisen wird das Produktportfolio zusammengestellt. Dieses Produktportfolio wird an den Einsatzplanoptimierer gesendet, damit dieser für den Produktportfoliooptimierer den Einsatzplan berechnen kann. Mit Hilfe dieser Einsatzplan, in Form einer Leistungskurv, kann der Produktportfoliooptimierer die Güte des Produktportfolios berechnen und letztendlich entscheiden, ob der Optimierungsvorgang fortgesetzt wird oder nicht. Wurde ein Produktportfolio erstellt, welches die Gütekriterien erfüllt, wird dieses an den Direktvermarkter gesendet. Die Gütekriterien beschreiben hierbei, ob beispielsweise der Einsatzplan überhaupt durchführbar ist oder ob das Produktportfolio Gewinn erzielt. Alle Gütekriterien werden im weiteren Verlauf der Dokumentation erläutert. Das Aktivitätsdiagramm kann der Abbildung 12 entnommen werden. Ein Übersicht aller PUCs befindet sich im Anhang D.1.6.

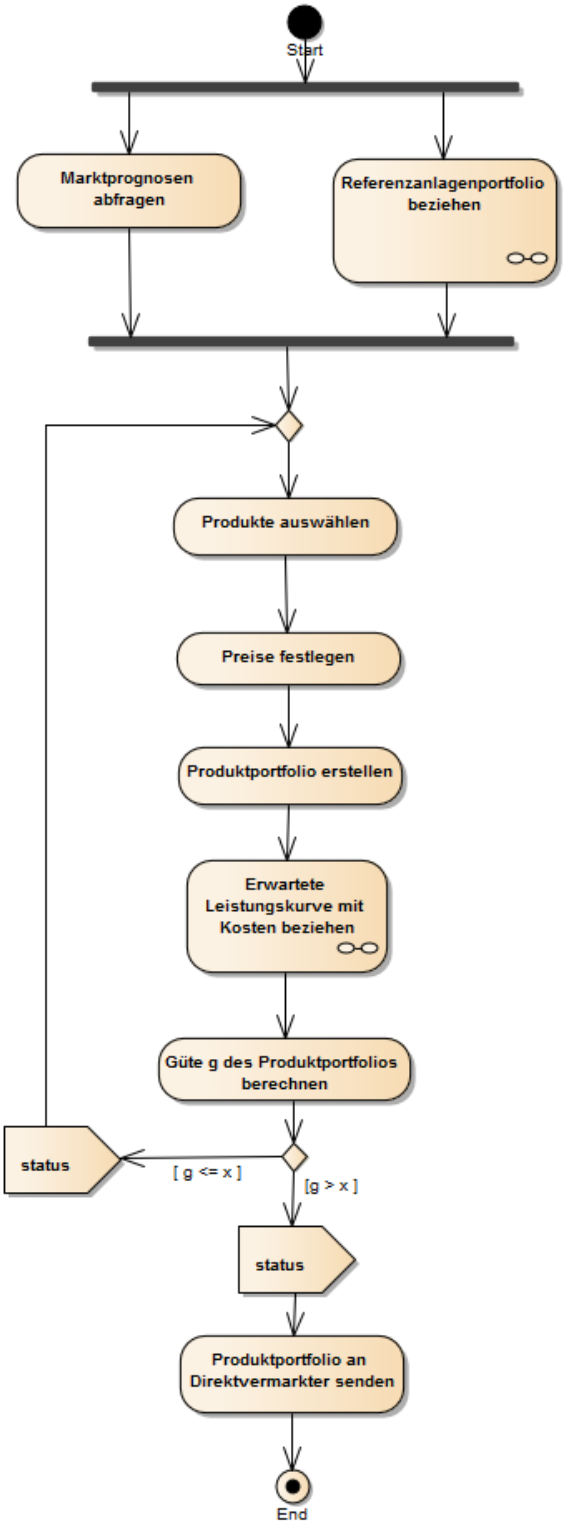


Abbildung 12: HLUC *Produktportfoliooptimierung* Aktivitätsdiagramm

Primary Use Cases des HLUCs *Produktportfoliooptierung*

ID	PPO-PUC1
Name	Erwartete Einsatzplankosten beziehen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Der Produktportfoliooptimierer verfügt über die zu erwartenden Kosten des Einsatzplan, der das Produktportfolio des aktuellen Optimierungsschritts umsetzt.
Kurz-beschreibung	Der Produktportfoliooptimierer sendet eine Anfrage an den Einsatzplanoptimierer. Der Einsatzplanoptimierer sendet die erwarteten Einsatzplankosten an den Produktportfoliooptimierer.
Fehlerfall	- Übertragungsfehler beim Senden der Einsatzplankosten.
Akteure	- Produktportfoliooptimierer - Einsatzplanoptimierer (s. Akteurabbildung: 101)
Vorbedingung	PPO-PUC5
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt die erwarteten Einsatzplankosten, um die Güte des Produktportfolios zu berechnen.
Ausgetauschte Information	- Leistungskurve mit Kosten (s. Aktivitätsdiagramm: 101)

ID	PPO-PUC2
Name	Marktprognose abfragen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Der Produktportfoliooptimierer verfügt über die Marktprognose.
Kurz-beschreibung	Der Produktportfoliooptimierer sendet eine Anfrage an das Marktmodul. Das Marktmodul sendet die Marktprognose für den gewünschten Tag an den Produktportfoliooptimierer.
Fehlerfall	- Übertragungsfehler beim Senden der Marktprognose.
Akteure	- Produktportfoliooptimierer - Marktmodul (s. Akteurabbildung: 104)
Vorbedingung	Verbindung zum Marktmodul
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt die Marktprognose, um die Produktportfoliooptimierung durchzuführen.
Ausgetauschte Information	- Marktprognose (s. Aktivitätsdiagramm: 104)

ID	PPO-PUC3
Name	Preise festlegen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	31.08.2015
Ziel	Die Verkaufspreise für die einzelnen Produkte sind festgelegt.
Kurz-beschreibung	Auf Basis der Marktprognosen legt der Produktportfoliooptimierer die Verkaufspreise für die ausgewählten Produkte fest.
Fehlerfall	- Verkaufspreise werden nicht oder fehlerhaft festgelegt.
Akteure	- Produktportfoliooptimierer (s. Akteurabbildung: 105)
Vorbedingung	PPO-PUC4
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt die Verkaufspreise, um das Produktportfolio erstellen und an den Direktvermarkter übergeben zu können.
Ausgetauschte Information	- Marktprognose -Referenzanlagenportfolio - Angebotspreise (s. Aktivitätsdiagramm: 105)

ID	PPO-PUC4
Name	Produkte auswählen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 156)
Erstellt am	29.06.2015
Ziel	Die Produkte, die in das Produktportfolio aufgenommen werden sollen, sind ausgewählt.
Kurz-beschreibung	Der Produktportfoliooptimierer analysiert die Marktprognose. Auf Basis der Analyse werden Produkte und deren zu verkaufende Energiemenge ausgewählt.
Fehlerfall	- Fehler bei Auswahl der Produkte. - Die zu verkaufende Energiemenge befindet sich nicht im gültigen Bereich.
Akteure	- Produktportfoliooptimierer (s. Akteurabbildung: 106)
Vorbedingung	PPO-PUC2, PPO-PUC7
Fachlicher Auslöser	Das Produktportfolio soll erstellt werden.
Ausgetauschte Information	- Marktprognose -Referenzanlagenportfolio - Produktliste (s. Aktivitätsdiagramm: 106)

ID	PPO-PUC5
Name	Produktportfolio erstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 156)
Erstellt am	31.08.2015
Ziel	Das Produktportfolio ist erstellt.
Kurz-beschreibung	Auf Basis der Produktauswahl und der festgelegten Preise wird ein Produkt-portfolio durch Zusammenführung dieser einzelnen Bestandteile erstellt.
Fehlerfall	- Produktportfolio ist nicht erstellt.
Akteure	- Produktportfoliooptimierer (s. Akteurabbildung: 107)
Vorbedingung	PPO-PUC3
Fachlicher Auslöser	Der Produktportfoliooptimierer soll die Kosten für das Produktportfolio be-rechnen.
Ausgetauschte Information	- Produktliste -Referenzanlagenportfolio - Produktportfolio (s. Aktivitätsdiagramm: 107)

ID	PPO-PUC6
Name	Produktportfolio an Direktvermarkter senden
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	29.06.2015
Ziel	Das Produktportfolio liegt dem Direktvermarkter vor.
Kurz-beschreibung	Der Produktportfoliooptimierer sendet das Produktportfolio an den Direkt-vermarkter.
Fehlerfall	- Übertragungsfehler. - Das Format des Produktportfolios ist falsch.
Akteure	- Produktportfoliooptimierer - Direktvermarkter (s. Akteurabbildung: 108)
Vorbedingung	PPO-PUC8 und die Güte des Produktportfolio ausreichend
Fachlicher Auslöser	Der VK-Admin möchte seine Energieprodukte am Markt verkaufen.
Ausgetauschte Information	- Produktportfolio (s. Aktivitätsdiagramm: 108)

ID	PPO-PUC7
Name	Referenzleistungskurve beziehen
Verbindlichkeit	muss
Status	verworfen (s. S. 175)
Erstellt am	01.07.2015
Ziel	Der Produktportfoliooptimierer verfügt über eine Referenzleistungskurve.
Kurz-beschreibung	Der Produktportfoliooptimierer sendet eine Anfrage an den Einsatzplanoptimierer. Der Einsatzplanoptimierer sendet eine Referenzleistungskurve an den Produktportfoliooptimierer.
Fehlerfall	- Referenzleistungskurve wird nicht gesendet.
Akteure	- Produktportfoliooptimierer - Einsatzplanoptimierer (s. Akteurabbildung: 109)
Vorbedingung	Verbindung zum Einsatzplanoptimierer
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt die Referenzleistungskurve.
Ausgetauschte Information	- Referenzanlagenportfolio (s. Aktivitätsdiagramm: 109)

ID	PPO-PUC8
Name	Güte g des Produktportfolios berechnen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 162)
Erstellt am	31.08.2015
Ziel	Die Güte g des Produktportfolios ist bestimmt.
Kurz-beschreibung	Der Produktportfoliooptimierer berechnet die Güte g (Gewinn) des Produktportfolios nach jedem Optimierungsschritt.
Fehlerfall	- Güte g des Produktportfolios ist nicht bestimmt.
Akteure	- Produktportfoliooptimierer - Einsatzplanoptimierer (s. Akteurabbildung: 110)
Vorbedingung	PPO-PUC1
Fachlicher Auslöser	Der Produktportfoliooptimierer benötigt die Güte des Produktportfolios, um zu entscheiden, ob die Optimierung des Produktportfolios weitergeführt werden soll.
Ausgetauschte Information	- Marktpreise - Umstaz - Einsatzplankosten - Gewinn (s. Aktivitätsdiagramm: 110)

High Level Use Case *Stammdaten der Anlagen bereitstellen* Das virtuelle Kraftwerk benötigt die Stammdaten der Anlagen, die in die Produktportfoliooptimierung einbezogen werden sollen. Die Stammdaten müssen von den Anlagenbetreibern zur Verfügung gestellt werden. Dieser Anwendungsfall wurde unabhängig davon modelliert, ob die Stammdaten elektronisch oder in Papierform vorliegen.

Primary Use Cases des HLUCs *Stammdaten der Anlagen bereitstellen* Aufgrund des geringen Umfangs dieses Anwendungsfalls wurde für diesen PUC nur die Anforderungstabelle erstellt und auf die weitere Modellierung im Enterprise Architect verzichtet.

ID	`Stamm-PUC1`
Name	Datenblätter der Anlagen erstellen
Verbindlichkeit	muss
Status	umgesetzt (s. S. 112)
Erstellt am	15.09.2015
Ziel	Für jede am VK registrierte Anlage liegt ein Datenblatt vor.
Kurz-beschreibung	Der Anlagenbetreiber erstellt für jede Anlage ein Datenblatt mit den Stammdaten der Anlage. Die Informationen dafür entnimmt er den vom Hersteller angegebenen technischen Daten.
Fehlerfall	- Für mindestens eine Anlage liegt kein Datenblatt vor.
Akteure	- Anlagenbetreiber
Vorbedingung	Für die Anlagen liegen technische Daten des Herstellers vor.
Fachlicher Auslöser	Die Anlage soll beim VK registriert werden.
Ausgetauschte Information	- Es werden keine Informationen ausgetauscht

Nach der Erstellung der Anforderungstabellen erfolgt im Weiteren als letzte Aufgabe in der Analysephase eine Betrachtung aller aufgeführten Akteure. Da diese in den noch zu betrachtenen Ebenen des SGAM-Modells benötigt werden, erscheint dieser Schritt sinnvoll, sowie wird dadurch eine Übersicht über alle Akteure gegeben.

4.4.3. Übersicht über die Akteure

Die folgende Tabelle zeigt für alle in der Analysephase ermittelten Akteure den Namen, den Typ und die Aufgabe. Der Typ der Akteure wurde bei der Erstellung des *Component Layers* (s. Abschnitt 4.5.1) festgelegt und stellt somit an dieser Stelle einen Vorgriff auf die Architekturphase dar. Jedem Akteur wird dafür ein Akteurtyp zugewiesen. Diese Akteurtypen sind durch SGAM selbst definiert. Da es sich bei allen aufgenommen Akteuren um keine physischen Geräte handelt, sondern lediglich um Software-Applikationen handeln, fallen alle Akteure heraus die sich mit der physischen Erzeugung, Weiterleitung, Umwandlung und Verteilung von Energie. Die für das Projekt interessanten Akteurtypen sind das Human-Machine Interface, welches beispielsweise eine GUI für eine Software darstellt, der Device, welche Akteure für den Netzschutz und das Management des Netzes darstellt,

sowie der Software-based Application Akteurtyp, welcher alle Arten von Software-Applikationen und Software-Systemen darstellt. Da der Akteurtyp des Devices jedoch Physische- bzw. Hardwarekomponenten darstellt fällt auch dieser aus der Betrachtung für das Projekt heraus. Bis zum Zeitpunkt des Zwischenberichts wurde des Weiteren GUI für die erstellte Software implementiert, sodass bis zum jetzigen Zeitpunkt der Akteurtyp des Human-Machine Interfaces noch keine Verwendung findet. Damit werden alle Akteure durch den Akteurtyp Software-based Application dargestellt. Begründet werden kann dies darüber, dass es sich bei allen Akteuren des Projektes um Software-Module handelt, die auf einer selbstausgewählten physischen Maschine laufen können. Die Akteure die weiterhin eine Datenbank beinhalten, werden durch zwei Software-based Application Akteurtypen dargestellt, da auch die Datenbanken keine physischen Maschinen.

Im Folgenden wird die komplette Akteurtabelle mit den Akteurtypen und Beschreibungen der Akteure dargestellt.

Akteure		
Akteurname	**Akteurtyp**	**Akteurbeschreibung**
Anlage	Software-based Application	Reale oder simulierte Anlage, auf der ein vom Einsatzplanoptimierer erstellter Fahrplan ausgeführt werden kann. Die Anlage kann weiterhin Flexibilitäten besitzen und erbringt über den Fahrplan Leistung.
Direktvermarkter	Software-based Application	Der Direktvermarkter erhält das erstellte Produktportfolio, um es an einer Strombörse zu vermarkten.
Einsatzplanoptimierer	Software-based Applicatio	Der Einsatzplanoptimierer erstellt auf Grundlage der bekannten Flexibilitäten der Anlagen und einer Lastkurve, die für die Umsetzung eines Produktportfolios erbracht werden muss, einen optimalen Einsatzplan.
Flexibilitätenmodul	Software-based Application	Das Flexibilitätenmodul bezieht Wetterdaten, Stammdaten und technische Restriktionen der Anlagen, diese werden an die Flexibilitätssimulationsumgebung weitergegeben. Auch startet das Modul den Vorgang in der Simulationsumgebung durch die die Flexibilitäten berechnet werden. Das Flexibilitätenmodul besitzt dabei eine Schnittstelle mit welcher der Benutzer interagieren kann, auch versendet es die Flexibilitäten an andere Module.
Flexibilitäts-simulations-umgebung	Software-based Application	Die Flexibilitätsumgebung beschreibt die Umgebung in der die Flexibilitäten berechnet werden. Sie bezieht dabei die Wetterdaten und Anlagenstammdaten aus dem Flexibilitätenmodul und versendet die berechneten Flexibilitäten.

Lokale Datenbank	Software-based Application, Datenbank	Die lokale Datenbank ist die zentrale Datenbank in der alle Wetterdaten, Marktdaten, Stammdaten und Flexibilitäten der Projektgruppe gespeichert werden. Diese besteht aus einem Datenbankmanagementsystem (Software-based Application) und der Datenbank selber. Das Datenbankmanagementsystem kommuniziert mit dem Marktmodul, VK-Controller und dem Wettermodul.
Marktdaten-provider	Software-based Application, Datenbank	Der Marktdatenprovider (Software-based Application) beschreibt einen externen Dienstleister, der Marktdaten der EEX-Spot bereitstellt. Dafür besitzt dieser eine eigene Datenbank. Die Kommunikation und der Datenaustausch erfolgt über das Marktmodul.
Marktmodul	Software-based Application	Das Marktmodul bezieht die Marktdaten vom Marktdatenprovider. Die erhaltenen Daten vergleicht dieses mit den Wetterdaten und historischen Marktdaten in der lokalen Datenbank. Dafür kommuniziert das Marktmodul mit dem Datenbankmanagementsystem. Auf Basis der bezogenen Daten werden Abhängigkeiten zwischen Wetter und Preisen im Marktmodul erstellt. Die neuen Daten werden dem Produktportfoliooptimierer zur Verfügung gestellt.
Produktportfolio-optimierer	Software-based Application	Der Produktportfoliooptimierer erstellt iterativ ein optimales Produktportolio für ein bestimmtes Anlagenportfolio. Er fragt dazu die Marktprognosen ab und bezieht die Referenzleistungskurve vom Einsatzplanoptimierer, um Produkte und deren Preise auswählen zu können. Das daraus erstellte Produktportfolio wird dem Einsatzplanoptimierer zur Verfügung gestellt. Anschließend werden die Einsatzplankosten für das erstellte Produktportfolio vom Einsatzplanoptimierer bezogen. Auf deren Basis berechnet der Produktportfoliooptimierer die Güte des Produktportfolios und entscheidet, ob das Produktportfolio im iterativen Prozess weiter verbessert werden soll. Entspricht das Produktportfolio der gewünschten Güte, sendet der Produktportfoliooptimierer dieses an einen Direktvermarkter.
VK-Admin	Software-based Application	Der VK-Admin startet den VK-Controller.
VK-Controller	Software-based Application	Der VK-Controller steuert die zentralen Funktionsaufrufe im VK. Er initialisiert Schnittstellen und Verbindungen und startet die Module.

Wetterdaten-provider	Software-based Applicati-on, Datenbank	Der Wetterdatenprovider (Software-based Application) beschreibt einen externen Dienstleister, der Wetterdaten bereitstellt. Dafür besitzt dieser eine eigene Datenbank. Die Kommunikation und der Datenaustausch erfolgt über das Wettermodul.
Wettermodul	Software-based Application	Aufgabe des Wettermoduls ist es, Wetterdaten (Wetterdaten und -prognosen) von einem externen Wetterdatenprovider zu beziehen, zu filtern, aufzubereiten und den anderen Modulen des Systems zur Verfügung zu stellen.

Nach der Erstellung der Business Use Cases auf Basis der Vision und der Aufgliederung in HLUC und PUC wurden Anforderungstabellen der einzelnen PUCs, sowie eine Akteurtabelle erstellt. Das dadurch entstandene Analyse-Dokument soll in einem nächsten Schritt als Grundlage für die Architekturphase dienen, in der, wie beschrieben, die weiteren Dimensionen des SGAM-Modells betrachtet werden.

4.5. SGAM: Architekturphase

Die Architekturphase beschreibt die zweite Phase der Anforderungserhebung in SGAM. Dabei sollen die bereits erstellten HLUC und PUC auf die technische und physische Ebene abgebildet werden. Hierfür werden *Component-*, *Information-* und *Communication Layer* benötigt. Diese Architekturphase aus SGAM stellt jedoch hier lediglich die grundlegende Basis für die Architektur des zu entwicklenden Systems dar. Das Ergebnis dieser Phase entspricht deshalb nicht der späteren Architektur, sondern dient für diese nur als Basis um die Architektur zu entwickeln (s. 4.1). Zudem wurde sich auch gegen eine nachträgliche Überarbeitung des SGAM entschieden, weil dies einen sehr hohen Überarbeitungsaufwand bedeuten würde und der Mehrwert für diese nicht gegeben war. In diesem Zusammenhang sind gerade die zu modellierenden Technologien usw. entweder noch nicht angegeben oder es wurde nur aufgezeigt, welche Möglichkeiten es dafür gibt.

Zunächst beginnt die Architekturphase mit der Erstellung des *Component Layers*, danach folgt der *Information Layer* und der *Communication Layer*. Das gesamte SGAM kann nach seinen Abschluss als ein Artefakt der Projektgruppe angesehen werden, aus dem die Basis der Architektur und die funktionalen Anforderungen hervorgehen.

4.5.1. Component Layer

Der *Component Layer* (*dt.* Komponentenebene) beschreibt die physische Verteilung der Komponenten im *Smart Grid*-Kontext. Dies beinhaltet alle verwendeten Komponenten, wie Applikationen, Elemente des Stromnetzes, Schutzelemente, Netzwerkinfrastruktur und jede Art von Computern und Geräten (devices), welche ihrem Akteur und dem Bereich im *Smart Grid*-Kontext des SGAM zugeordnet werden [Gro13]. Dadurch kann beschrieben werden, welche Art von physischen Geräten im Projekt genutzt werden und wo sie im Energiekontext angesiedelt sind. Im Folgenden sollen nun den

identifizierten Akteuren aus der Analysephase (Unterunterabschnitt 4.4.3) ihre physischen oder virtuellen Komponenten zugewiesen und im Kontext des *Smart Grids* auf die verschiedenen Domänen und Zonen verteilt werden. Dies beschreibt den ersten Schritt der Architekturphase und somit den *Component Layer* des SGAM. Die Komponenten beinhalten dabei bereits den in der Akteurtabelle enthaltenen Akteurtyp.

Actor Mapping Model Zunächst wurde jedem Akteur eine physische oder virtuelle Komponente zugewiesen. Diese Verteilung kann der Abbildung 13 entnommen werden.

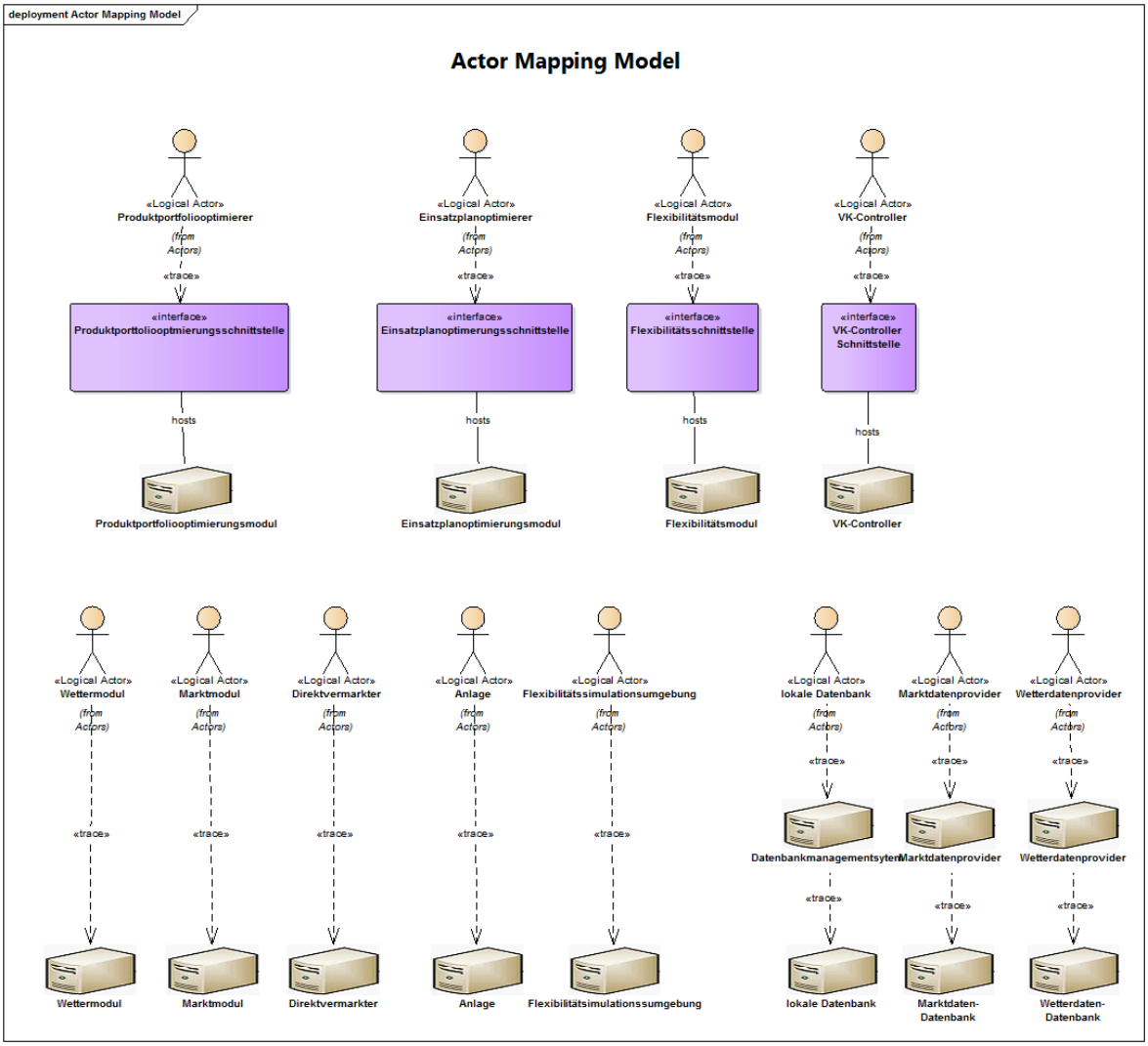


Abbildung 13: Übersicht der Actors

Die Abbildung 13 zeigt, dass jedem Akteur der Akteurtyp *Software-based Application* zugeordnet wurde. Der Akteurtyp wird dabei mit dem Symbol eines Computers dargestellt (s. 13). Wie bereits in der Beschreibung der Akteurtabelle aufgezeigt, wurde der Akteurtyp *Software-based Application* ausgewählt, da es sich bei allen Komponenten des Projektes um Softwaremodule handelt. In diesem Anwendungsfall gibt es also keine Komponente, die beispielsweise eine physische Anlage wäre. Da selbst die Anlagen nur simuliert werden, wird auch hier nur der *Software-based Application-*

Akteurtyp benötigt. Auch bei den Datenbanken wurde, wie im vorherigen Kapitel erwähnt, nur der *Software-based Application* Akteurtyp ausgewählt, da auch bei diesen nicht unbedingt die physische Maschine bekannt ist. Als Besonderheit besitzen die Akteure *VK-Controller*, *Flexibilitätenmodul*, *Produktportfoliooptimier* und *Einsatzplanoptimierer* Schnittstellen neben den *Software-based Application*. Diese werden für die Kommunikation mit den *Software-based Application* verwendet und werden auch nur für diese benötigt. Jede *Software-based Application* wurde des Weiteren so konzipiert, dass diese als eigenständiges Modul funktioniert und somit alle Module auf unterschiedlichen oder dergleichen physische Maschine laufen können. Die Begründung und Entscheidung für die Modularität der einzelnen Softwarekomponenten, kann dem Unterkapitel Unterabschnitt 6.1 entnommen werden.

Smart Grid Plane Nachdem jedem Akteur seine physischen oder virtuellen Komponenten zugeordnet wurden, ist die weitere Aufgabe, die Komponenten nun in die Zonen und Domänen des SGAM aufzuteilen. Dies geschieht für jeden einzelnen HLUC separat, sodass im Falle der Projektgruppe sechs Abbildungen entstehen. Diese können dem Anhang D.3 entnommen werden. Für die Erläuterung der Position der Komponenten wurde des Weiteren eine Abbildung der Gesamtübersicht aller Komponenten in dem Gitternetz aus Domänen und Zonen erstellt. Diese kann der Abbildung 126 entnommen werden.

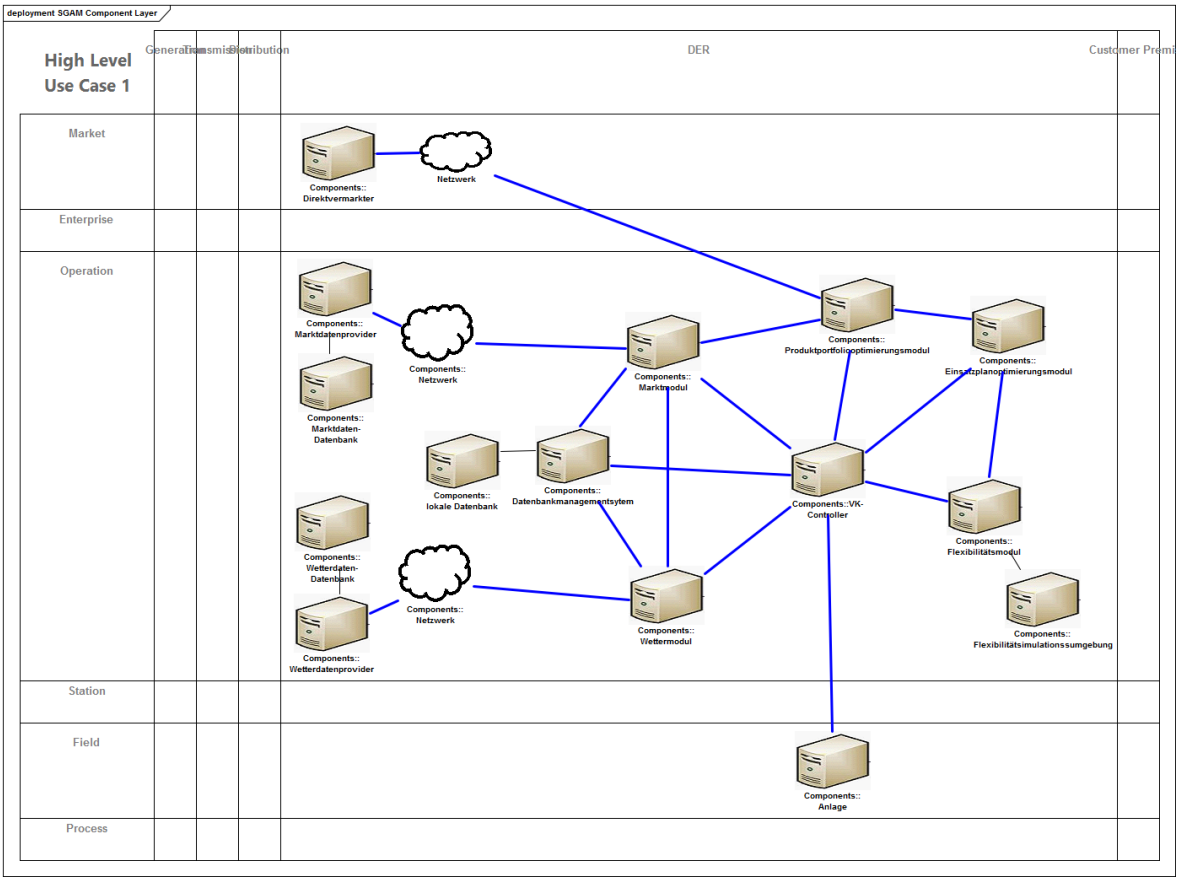


Abbildung 14: Übersicht aller Komponenten in der Ebene

Die Verbindungslinien stellen die Verbindungen zwischen den einzelnen Komponenten dar. Dies bedeutet wiederum, dass auch nur die verbundenen Komponenten miteinander agieren können. Welche Komponenten miteinander verbunden werden mussten, wurde aus den Anforderungstabellen der Analysephase entnommen.

Wie die Abbildung weiterhin darstellt, befinden sich alle Komponenten in der Domäne *Distributed Energy Ressource* (*DER*). Begründet werden kann dies damit, dass über den Modulen übergeordnet das virtuelle Kraftwerk steht und dieses virtuelle dezentrale Energiekomponenten zusammenfasst. Da auch die simulierten Anlagen als dezentrale Energiesysteme simuliert werden, werden auch diese in die Domäne DER gesetzt. Wiederum benötigen virtuelle Kraftwerke keine physische Verbindungen zwischen ihren Komponenten, wie beispielsweise Stromkabel, sodass die Domänen *Distribution* und *Transmission* nicht benötigt werden. Da auch keine Endverbraucher, wie beispielsweise Haushalte, direkt mit einbezogen werden, wird auch die Domäne *Customer Premises* im Kontext der Projektgruppe nicht verwendet. Von allen aufgezeigten Domänen ist des Weiteren eine ausführlichere Dokumentation in dem beigefügten Seminarband G enthalten. Auch die Verteilung der Komponenten in die unterschiedlichen Zonen kann der Abbildung 126 entnommen werden. Diese Aufteilung wird vorgenommen, da die Zonen die unterschiedlichen hierarchischen Ebenen der Energieversorgung darstellten. In der Abbildung ist zu erkennen, dass die meisten Komponenten in der Zone *Operation* liegen. Diese beschreiben Systeme für Stromversorgungssteuerungsoperationen, wie beispielsweise Energiemanagement Systeme. Bis auf die Anlagen, den Produktportfoliooptimierer und den Direktvermarkter werden alle Komponenten benötigt, um Prognosen für erzeugte Energie zu erstellen, diese an die Anlagen zu senden oder Prognosen für den Markt damit zu kreieren. Damit kann begründet werden, warum die Komponenten in dieser Zone aufzufinden sind. Physische Anlagen werden dagegen in die Zone *Process* eingeordnet. Da es sich im Rahmen der Projektgruppe jedoch nur um eine Simulation handelt, wurde die Komponente der Anlage in die Zone *Fields* gesetzt. Diese beinhaltet eigentlich nur den Controller, welcher die Steuerung und die Kommunikation mit der Anlage übernimmt. Da die Simulation genau solche Prozessdaten erzeugt, wurde die Komponente auch hierhin zugeordnet. Zudem beinhaltet diese Ebene alle Komponenten zur Überwachung und zum Schutz von Anlagen und Kabeln. Es wurde daher beschlossen das Anlagenmodell auf diese Ebene zu setzten, sodass auch die Domäne *Generation*, in der die physischen Anlagen dargestellt werden, nicht betrachtet werden muss. Der Produktportfoliooptimierer ist die Komponente, die das fertige Produktportfolio erzeugt und an den Direktvermarkter weitergibt. Somit ist es auch die Komponente, die ein kommerzielles Produkt entwickelt. Diese Komponenten werden deshalb in der Zone *Enterprise* eingegliedert. Die letzte Komponente ist der bereits angesprochene Direktvermarkter. Dieser handelt mit den Portfolios direkt am Markt, weshalb dieser auch der Zone *Market* zugeordnet werden kann. Auch für die Zonen ist im Seminarband eine ausführlichere Beschreibung vorhanden.

Nach der Erstellung des *Component Layers* müssen in der SGAM-Architekturphase nun der *Information-* und *Communication Layer* entwickelt werden. Dafür werden unter anderem die erzeugten Komponenten benötigt.

4.5.2. Information Layer

Im *Information Layer* (*dt.* Informationsebene) werden die Informationen, die verwendet werden, und der Austausch dieser zwischen den Funktionen, Services und Komponenten beschrieben. Die Informationen werden dabei über *Information Objects* (*dt.* Informations Objekte) und der Austausch der Informationen über standardisierte Datenmodelle (Canonical Data Model) beschrieben. Zusammen beschreiben diese die Kommunikationssemantik für die Funktionen und Services und erlauben so einen interoperablen Informationsaustausch über Kommunikationsmittel [Gro13].

Die Modellierung des *Information Layers* erfolgt in drei Schritten, die für jeden HLUC getrennt ausgeführt und im Folgenden beispielhaft am HLUC *Wettervorhersage* gezeigt werden. Bei den Phasen handelt es sich um den *Business Context View*, das *Standard and Information Object Mapping* und den *Canonical Data Model View*. Die weiteren Modelle für die übrigen HLUC befinden sich im Anhang D.2.

Business Context View Im *Business Context View* wird betrachtet, zwischen welchen Komponenten Datenflüsse bestehen und welche Informationen dabei ausgetauscht werden. Diese Informationen werden durch Informationsobjekte modelliert, die den Datenflüssen zugeordnet sind.

Im Beispiel des HLUC *Wettervorhersage* ist es vorgesehen, dass das Wettermodul Wetterdaten und eine Wetterprognose von einem Wetterdatenanbieter bezieht. Wie in Abbildung 15 zu sehen, ist dieser Datenfluss als Pfeil modelliert worden, der mit den Informationsobjekten *Wetterdaten* und *Wetterprognose* belegt ist. Damit das Wettermodul Daten vom Wetterdatenprovider erhalten kann, muss es Login-Daten senden. Dies wird durch den mit dem Informationsobjekt *Login-Daten* belegten Pfeil vom Wettermodul zum Wetterdatenanbieter modelliert.

Nach demselben Prinzip wurden auch die übrigen Verbindungen modelliert: das Wettermodul speichert die erhaltenen Daten in eine Datenbank und stellt dem Marktmodul die Wetterdaten zur Verfügung.

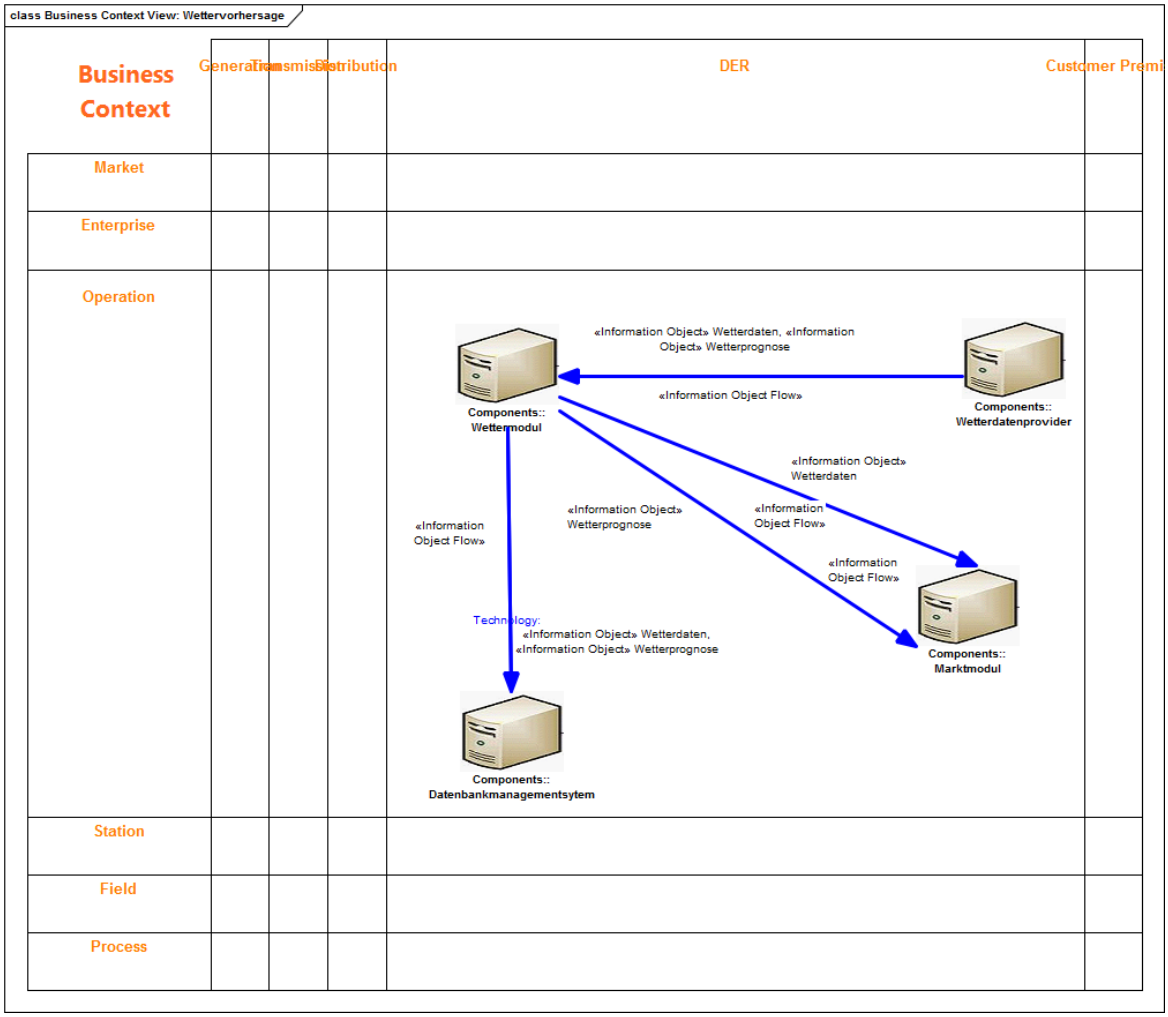


Abbildung 15: Business Context View für HLUC *Wettervorhersage*

Standard and Information Object Mapping Im zweiten Schritt des *Information Layers*, dem *Standard and Information Object Mapping*, wird jedem zuvor modellierten Informationsobjekt ein Datenmodellstandard zugeordnet. Dabei kann es vorkommen, dass einem Informationsobjekt verschiedene Datenmodellstandards zugeordnet sind und umgekehrt. Im HLUC *Wettervorhersage* sollen, wie in Abbildung 16 zu sehen, beispielsweise die Wetterdaten und -prognose, die das Wettermodul vom Wetterdatenanbieter erhält, dem Standard Weather Information Exchange Model (WXXM) entsprechen. Dieser Standard besitzt eine spezielle Form des *xml*-Formats, mit der die weitere Arbeit im Projekt schwieriger erscheint. Deshalb wird diese über einen Parser einmalig in Java-Objekte umgewandelt, die mit dem eigenen Standard *Mein Wetter Standard* bezeichnet werden. Diese können einfacher von dem Wetter- und Marktmodul verwendet werden und sind auch so in der Datenbank gespeichert.

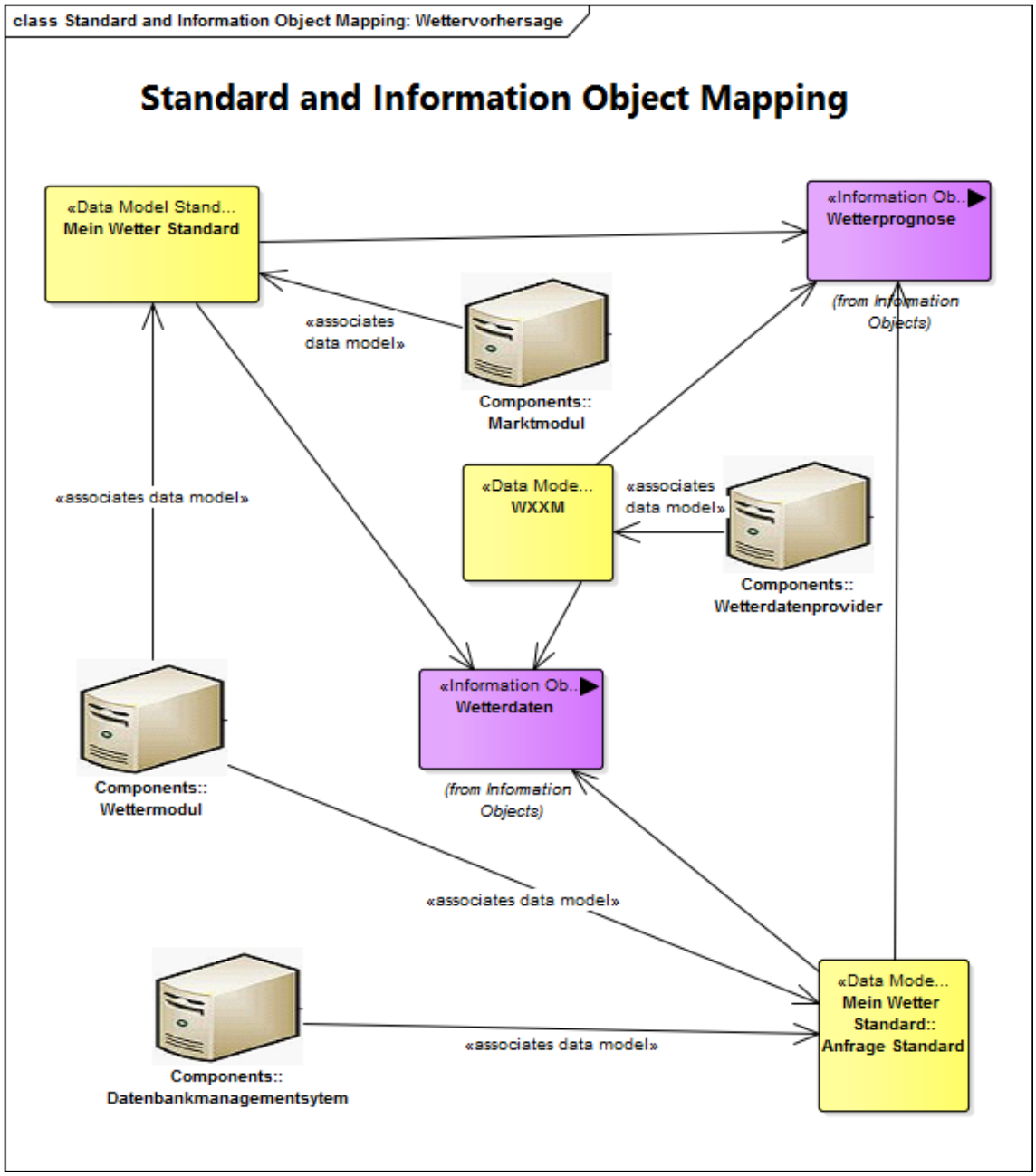


Abbildung 16: Standard and Information Object Mapping für HLUC *Wettervorhersage*

In der Modellierung aller HLUC wurden zunächst für alle VK-internen Datenflüsse eigene Datenmodellstandards angenommen (*Mein . . . Standard*), da zum Abschluss des SGAM noch keine Entscheidung über das verwendete interne Datenmodell getroffen wurde. Die Entscheidung über das genutzte interne Datenmodell kann dem Abschnitt 6.2.1 entnommen werden. Neben den Datenmodellstandards (WXXM bzw. HTML) für die Schnittstellen nach außen zum Wetter- und Marktdatenprovider muss des Weiteren die externe Kommunikation zu den Anlagen betrachtet werden. Auch wenn diese im Beispiel des HLUC *Wettervorhersage* nicht betrachtet wird, ist diese von sehr großer Bedeutung für das Projekt, weshalb auch der dazu genutzte Standard dargestellt werden soll. Auch die Entscheidung für das Datenmodell für die externe Kommunikation mit den Anlagen via Common Information

Model (CIM) oder *IEC 61850* wurde zum Abschluss der SGAM-Phase noch nicht getroffen. Deshalb wurde auch für diesen Fall ein eigener Standard *Mein Anlagenmodell* (s. Abbildung 118) angenommen. Die Entscheidung und Verwendung des Datenmodells für die Anlagenkommunikation kann dem Abschnitt 6.2.3 entnommen werden.

Canonical Data Model View Im letzten Schritt des *Information Layers* werden die Datenmodellstandards in die von den Domänen und Zonen aufgespannte *Smart Grid Plane* eingetragen. Dabei wird berücksichtigt, in welchen Bereichen sich die Komponenten befinden, die die entsprechenden Informationsobjekte verwenden. Im vorliegenden Beispiel liegen alle Komponenten und somit auch alle Datenmodellstandards in der Schnittstelle von *Operation* und *DER* (s. Abbildung 17).

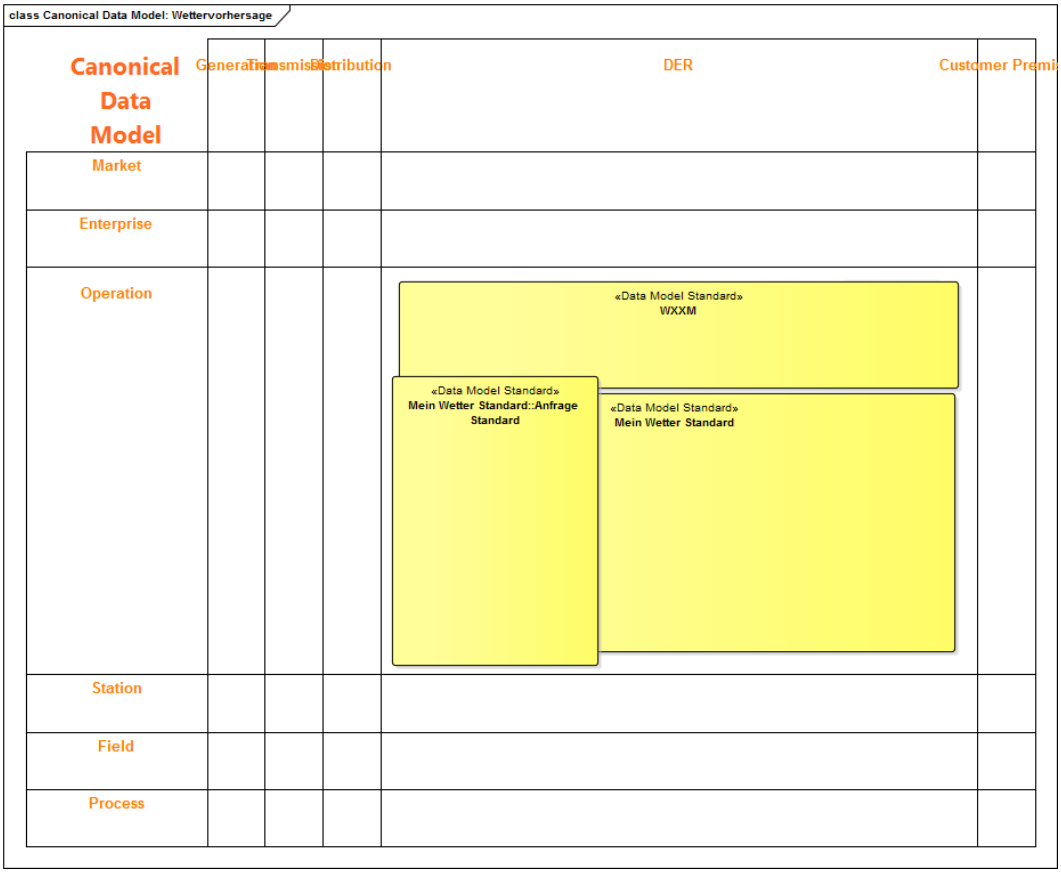


Abbildung 17: Canonical Data Model View für HLUC *Wettervorhersage*

4.5.3. Communication Layer

In der letzten Ebene des SGAM werden im *Communication Layer* (*dt.* Kommunikationsebene) die verwendeten Protokolle und Mechanismen für den interoperablen Austausch der Information zwischen den Komponenten beschrieben. Diese unterliegen dabei dem Kontext der *Use Cases*, Funktionen oder Services und den Informationsobjekten oder Datenmodellen [Gro13].

Für das Projekt werden im *Communication Layer* alle genutzten Technologien und Protokolle zur Kommunikation zwischen den einzelnen Modulen dargestellt, die bis zum Abschluss des SGAM-Abschnittes bereits entschieden wurden. Eine solche Übersicht wurde für jeden HLUC aufgestellt

und wird hier beispielhaft am HLUC *Wettervorhersage* aufgezeigt. Alle weiteren Abbildungen des *Communication Layer* für die HLUC kann dem Anhang D.4 entnommen werden. Für die Erstellung jedes *Communication Layers* wurden im Projektkontext alle im HLUC beteiligten Komponenten in ihrem jeweiligen Schichtenkontext eingebettet und eingezeichnet. Anschließend wurden die Kommunikationsschnittstellen zwischen denselben mit den dafür verwendeten Protokollen und Technologien beschriftet (s. beispielhaft 18).

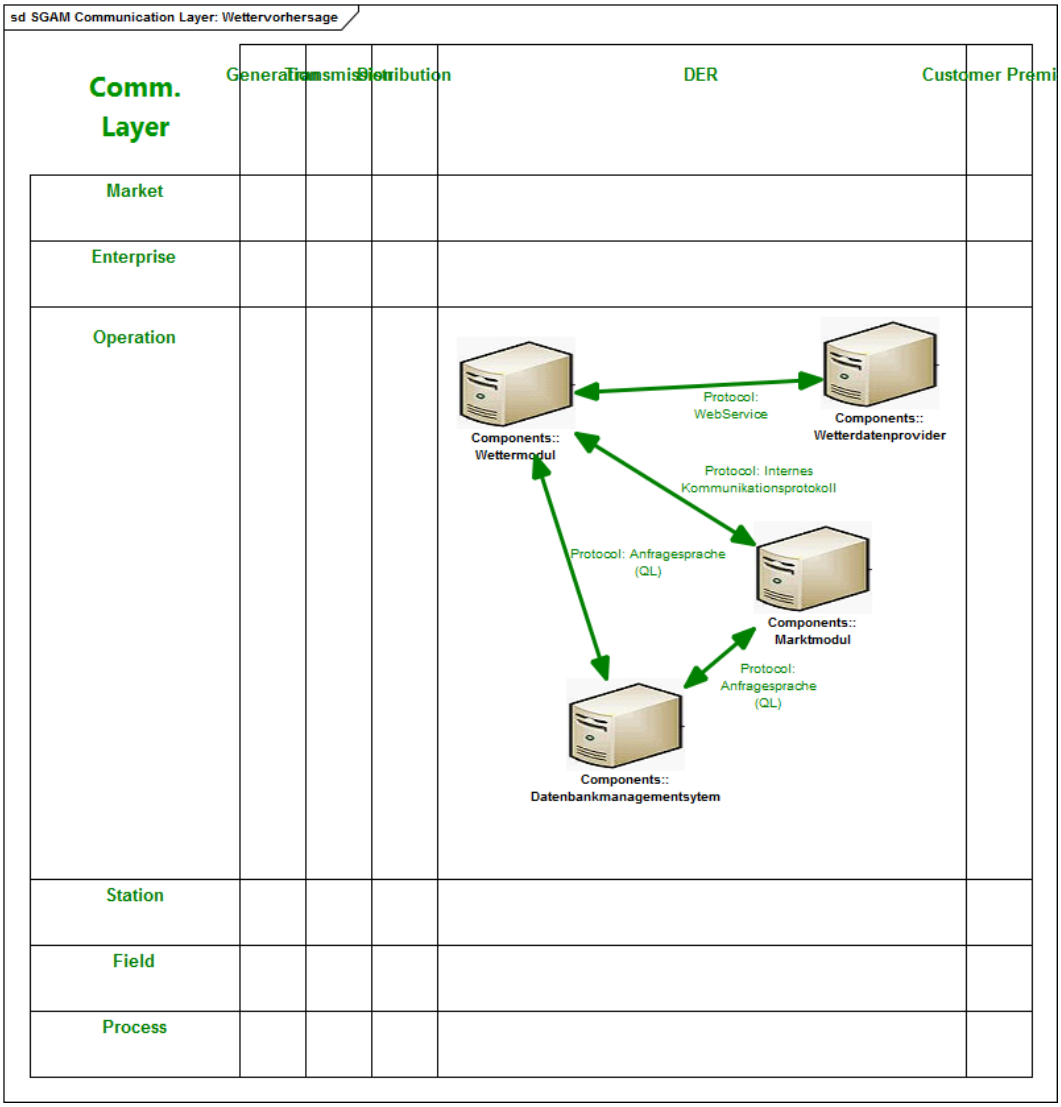


Abbildung 18: Communication Layer für HLUC *Wettervorhersage*

Im Weiteren werden nun alle benötigten Kommunikationsprotokolle und Technologien für die identifizierten Kommunikationsschnittstellen des Projektes betrachtet. Da bereits angedeutet wurde, dass zum Abschluss der SGAM-Phase noch keine Entscheidung in einigen Bereichen getroffen wurden, sind die Angaben auch im *Communication Layer* noch keine Endgültigen und wurden zumeist mit Platzhalterstandards modelliert. Zudem ist es bei der Entscheidung für ein Protokoll wichtig zu betrachten, welches Datenmodell für welche Kommunikation verwendet werden soll. Die Entscheidung

für ein standardisiertes Kommunikationsprotokoll, für jeden einzelnen Fall, wurde dann in anderen Kapiteln getroffen. Es wurden folgende Kommunikationsprotokolle in SGAM modelliert:

- WebServices
- Ethernet
- Anfragesprache
- internes Kommunikationsprotokoll
- externes Kommunikationsprotokoll

WebServices Das Kommunikationsprotokoll *WebServices* wurde für die identifizierten Kommunikationsschnittstellen von *Powder* zu den Direktvermarkter und den Wetter- und Marktdatenprovider modelliert. Das *WebServices* übernimmt dabei einen Platzhalter für die im späteren ausgewählten Standard und die dazu nötige Technologie. Für beide Schnittstellen, sind auch andere Möglichkeiten für die Umsetzung der Anforderungen möglich, wie ein *Web-Scraper* oder eine grafische Benutzerschnittstelle für den Direktvermarkter. Die jeweilige Entscheidung kann den Abschnitten 6.7, 6.6 und 6 entnommen werden.

Ethernet Die Technologie *Ethernet* wurde auch für die Kommunikation und den Datenaustausch zu den Markt- und Wetterdatenprovider modelliert. Auch hier dient dies nur als Platzhalter, da die Entscheidung noch nicht getroffen wurde. Da die Provider jedoch ihre Daten auf Webseiten anbieten, wird sich die ausgewählte Möglichkeit für die Kommunikation auf Webstandards beziehen müssen. Die Entscheidung kann in den Abschnitten 6.7 und 6.6 betrachtet werden.

Anfragesprache Die *Anfragesprache* (*engl.* query language) dient als Platzhalter für das Kommunikationsprotokoll, welches für die Kommunikation mit der eigenen Datenbank verwendet werden soll. Dabei konnte der Platzhalter bereits auf eine Anfragesprache reduziert werden, da zum Zeitpunkt des SGAM-Abschlusses die Verwendung einer relationale Datenbank entschieden wurde 5.2. Die Anfragesprache wird dabei zur Kommunikation mit dem Datenbankmanager verwendet. Dieser befindet sich vor der Datenbank und übernimmt alle Anfrage und Datenaustausch-Vorgänge. Die Entscheidung kann dem Abschnitt 6.5 entnommen werden, welcher auch die Entscheidung über die gewählte Technologie enthält.

Internes Kommunikationsprotokoll Dieses Protokoll wird für alle internen Kommunikationen benötigt, vor allem zwischen den Modulen. Daher ist dies von zentraler Bedeutung für *Powder*. Da auch hier die Entscheidung über ein Protokoll und die notwendige Technologie zum Abschluss von SGAM noch nicht getroffen wurde, wurde auch hier ein Platzhalter modelliert. Der Platzhalter wurde dafür *internes Kommunikationsprotokoll* genannt, die Entscheidung und die Implementierung des verwendeten internen Kommunikationsservice und den dafür benötigten Protokollen kann dem Abschnitt 6.2.1 entnommen werden.

Externes Kommunikationsprotokoll Die Schnittstelle zu den Anlagen ist die letzte notwendige Kommunikationsschnittstelle. Dafür wird ebenso, wie für alle anderen Schnittstellen, ein Kommunikationsprotokoll benötigt. Da viele unterschiedliche Anlagentypen angesprochen werden sollen, ist auch hier ein Standard notwendig. Da diese Entscheidung einen hohen Rechercheaufwand bedeutet, wurde auch diese Entscheidung erst später getroffen. Eine ausführliche Beschreibung aller Möglichkeiten und deren Technologien ist in dem Abschnitt 6.2.3 dargestellt.

Mit Abschluss des *Communication Layers* ist auch die Architekturphase des SGAM abgeschlossen, was die Fertigstellung des SGAM für das Projekt bedeutet. Dieses entstandene Artefakt wurde nun im Weiteren zusammen mit den speziellen und den funktionalen Anforderungen aus der Analysephase für die Entwurfs- und Implementierungsphase verwendet. Dieser Vorgang wird im Kapitel Architektur 6 beschrieben.

5. Technologien

In diesem Kapitel werden verschiedene Technologien vorgestellt, die zur Erstellung des Programmcodes, zur Aufrechterhaltung der Codequalität und für den Entwurf der Architektur verwendet wurden. Der letzte Abschnitt enthält eine Evaluation der zuvor vorgestellten Technologien.

5.1. Programmiersprache

In diesem Abschnitt wird erläutert, weshalb die Wahl der Programmiersprache auf Java fiel. Zunächst haben sich drei in frage kommende Programmiersprachen herausgestellt, mit denen mindestens eine Teilmenge der Projektgruppenmitglieder bereits Erfahrung gesammelt hatte: C++, Python und Java.

C++ ist für größere Projekte geeignet und bietet die Möglichkeit, zum Teil sehr elegante Lösungen mit Hilfe von Zeigern zu formulieren. Das daraus resultierende Programm kann sehr performant sein, da es, anders als die beiden anderen Optionen, nicht in einer virtuellen Maschine ausgeführt wird. Allerdings benötigt C++ eine höhere Einarbeitungszeit als Java oder Python und gerade für C++-Neulinge gibt es viele Möglichkeiten, Fehler zu verursachen. Zusammen mit der Tatsache, dass nur wenige Mitglieder der Projektgruppe überhaupt umfangreichere Erfahrungen mit dieser Programmiersprache gesammelt hatten, bedeutete dies eine Entscheidung gegen C++.

Die Skriptsprache Python ist für kleine Projekte sehr gut geeignet, kann aber auch zur Entwicklung größerer Projekte verwendet werden. Die Stärken von Python liegen in der Einfachheit der Programmierung. Viele Funktionen und Konstrukte lassen sich dank einfacher Syntax sehr kurz und elegant formulieren, sodass der Quellcode eines Python-Programms bis zu einem Drittel kürzer sein kann, als zum Beispiel ein vergleichbarer Java-Code. Auch das Importieren von Modulen ist in der Regel unkompliziert. Als Nachteil ist lediglich aufzuführen, dass noch nicht alle Projektgruppenmitglieder Erfahrung mit Python sammeln konnten.

Für Java spricht, dass es für größere Projekte sehr gut geeignet ist und sehr viele Frameworks und Plug-ins existieren. Im Gegensatz zu C++ und wie bei Python gibt es wenige potenzielle Fallstricke. Der wohl größte Vorteil war allerdings, dass sich alle Projektgruppenmitglieder bereits mit Java auskannten und sich dementsprechend niemand neu einarbeiten musste.

Anhand dieser Argumente wurde Java als Programmiersprache ausgewählt.

5.2. Datenbanken

In diesem Unterabschnitt werden Technologien vorgestellt, welche im Zusammenhang mit der persistenten Speicherung von Daten in der Datenbank eingesetzt wurden. Dazu gehören Hibernate, MySQL und H2. Diese werden benötigt, um Markt- und Wetterdaten abzuspeichern und wieder abrufen zu können.

Hibernate In Unterunterabschnitt 6.5.1 wird erläutert, warum für die persistente Speicherung der Daten in *Powder* eine relationale Datenbank verwendet wird. Mit Hilfe von Hibernate lassen sich Java-Objekte auf einfache Weise in einer relationalen Datenbank speichern. Dafür muss die Java-Klasse in der Hibernate-Konfigurationsdatei `hibernate.cfg.xml` eingebunden und sowohl die gesamte Klasse, als auch die einzelnen Felder annotiert werden, wie in Code-Ausschnitt 1 gezeigt:

Code-Ausschnitt 1: Hibernate Annotationen Beispiel

```
1  // Diese Klasse wird in der Datenbank unter der Tabelle "DB_Persons" gespeichert
2  @Entity(name = "DB_Persons")
3  public class Person {
4
5      @Id     // Dieses Feld repräsentiert die ID
6      @Column    // Dieses Feld wird mit in die Datenbank aufgenommen
7      @GeneratedValue  // Dieser Wert wird automatisch generiert
8      private long id;
9
10     // Dieses Feld wird in der Tabelle unter der Spalte "person_name" gespeichert
11     @Column(name = "person_name")
12     private String name;
13 }
```

Datentypen werden automatisch auf SQL-Datentypen abgebildet und referenzierte Objekte in eigene Tabellen gespeichert. Dabei bleibt die Referenz nach dem Auslesen erhalten. Auch Listen, Mengen und Schlüssel-Werte-Paare werden von Hibernate unterstützt. Dadurch wird der Aufwand abgenommen, der bei der Erstellung von SQL-Anfragen für jede Klasse und für jede *create, read, update* und *delete* (CRUD) Operation entstehen würde. Zudem werden Änderungen in der Klasse, wie neu hinzugefügte oder entfernte Felder, in der entsprechenden Datenbanktabelle automatisch angepasst. Hibernate vereinfacht die Handhabung mit relationalen Datenbanken um ein Vielfaches. Alle Funktionalitäten sind in der Hibernate-Dokumentation[3] zu finden.

MySQL Für die persistente Speicherung von Daten wurde in *Powder* die relationale Datenbank MySQL verwendet. Gründe dafür waren zum einen die quelloffene Implementierung, die im Gegensatz zu Oracles SQL kostenfrei verwendet werden kann. Zum anderen ist MySQL weit verbreitet und war allen *Powder*-Mitgliedern bekannt. Somit erübrigte sich die Einarbeitung in diese Technologie und die Datenbank konnte in kurzer Zeit aufgesetzt werden.

H2 Damit *Powder* auch lokal und ohne die Installation komplexer Datenbanksysteme ausführbar ist, kann neben MySQL auch die schnelle, eingebettete Datenbank H2 genutzt werden. Die Wahl fiel auf H2, da diese ohne weitere Konfiguration mit Hibernate funktionierte. Da H2 in Java implementiert ist, war für die Einrichtung lediglich eine Maven Abhängigkeit hinzuzufügen. H2 ist quelloffen und bietet SQL JDBC-Treiber.

5.3. Lombok

Das Projekt Lombok ist eine IDE-Erweiterung, welche spezielle Annotationen im Java-Code erkennt und kompiliert. Auf diese Weise müssen strukturell gleich aufgebaute Methoden wie Getter, Setter, `toString()`, `equals()` und `hashCode()` nicht für jede Klasse neu geschrieben werden. Mit der

[3] `http://hibernate.org/orm/documentation`

Annotation `@Data` über einer Klasse kann die IDE die genannten Methoden benutzen, obwohl sie nicht explizit geschrieben wurden.

Entsprechend der Code-Konvention der Projektgruppe beginnen alle Klassenvariablen mit einem *m*. Damit dieses jedoch nicht im Getter enthalten ist, muss zusätzlich die Annotation `@Accessors (prefix = "m")` genutzt werden. So wird statt dem Getter `getMAge()` der verständlichere Getter `getAge()` generiert.

```
@Data
@Accessors(prefix = "m")
class Person {
    int mAge = 10;
}
```

Lombok erzeugt außerdem automatisch die `toString`-Methode für alle Objekte, was beim Debuggen und Loggen sehr nützlich ist. Insgesamt macht Lombok den Code deutlich kürzer. Da es aktiv auf der zum Mitmachen offenen Versionsverwaltung-Webseite GitHub weiterentwickelt wird und dort auch viel Zuspruch findet, spricht für die Projektgruppe nichts gegen die Verwendung. Zudem kann das eingebaute Modul *delombok* ausgeführt werden, welches das Projekt kopiert und in normalen Java-Code konvertiert. Die Installation in IntelliJ erfolgt in den Einstellungen, während für Eclipse einmalig eine Jar-Datei ausgeführt werden muss.

Als Alternative zu Gettern und Settern könnten auch *public fields* genutzt werden. Gegenüber diesen haben Getter und Setter aber gravierende Vorteile:

- Der Zugriff wird über Schnittstellen gekapselt. Dadurch können Seiteneffekte zentral im Getter oder Setter implementiert werden.
- Lesezugriff kann ohne Schreibberechtigung gewährt werden.

Aufgrund dieser Vorteile hat sich die Projektgruppe gegen *public fields* entschieden.

5.4. Maven

Maven ist ein Framework, welches die Softwareentwicklung beim Kompilieren, Testen und Packen unterstützt. Es wurde eingesetzt, um das Einbinden von Abhängigkeiten und das Erstellen von Jar-Dateien zu vereinfachen. Dabei gibt Maven eine einheitliche Ordnerstruktur vor, welche folgendermaßen aufgebaut ist:

src		Gesamtes Modul
	main	Produktive Dateien
	java	Produktiver Java Sourcecode
	ressources	Dateien, die vom Sourcecode verwendet werden (Bilder, etc)
	test	Test Dateien
	java	Java Testfälle
	ressources	Dateien, die von Testfällen verwendet werden (CSV, etc)

Wird ein Maven-Projekt kompiliert, so werden in der Standardkonfiguration anschließend alle Testfälle ausgeführt. Laufen alle Testfälle erfolgreich ab, so schließt Maven den Build erfolgreich ab.

Dadurch wird eine Jar-Datei erstellt, welche die kompilierten Sources enthält. Ansonsten wird eine Fehlermeldung ausgegeben. Dieser Mechanismus wirkt sich positiv auf die Softwarequalität aus.

Zudem ist es mit Hilfe von Maven möglich, Abhängigkeiten zu anderen Projekten zu erzeugen. So können externe Bibliotheken leicht in das eigene Projekt eingebunden werden. Diese Abhängigkeiten und weitere Konfigurationen können in jedem Modul in der standardmäßig vorhandenen Datei pom.xml eingetragen werden.

Bei großen Softwareprojekten ist es oft hilfreich, das gesamte System in Module aufzuteilen. Durch Maven können diese Module als eigenständige Projekte betrachtet werden. Abhängigkeiten unter diesen Modulen können in der bereits erwähnten pom.xml eingetragen werden. Dadurch kann parallel an verschiedenen Modulen gearbeitet werden, ohne dass sich die gleichzeitige Bearbeitung gegenseitig behindert. Alle Funktionalitäten von Maven sind in der Maven-Dokumentation[4] zu finden.

Andere Build-Tools, wie Ant oder Gradle, setzten weniger Wert auf standardisierte Strukturen und geben den Entwicklern mehr Freiräume in Bezug auf Konfigurationsmöglichkeiten. So können Build-Skripte in Ant und Gradle wesentlich leichter eingebunden werden als in Maven. Für *Powder* wurde dennoch Maven verwendet, da keine eigenen Build-Skripte benötigt wurden. Zudem hatten viele Mitglieder bereits praktische Erfahrungen mit Maven. Die anderen beiden Build-Tools waren dagegen kaum bekannt.

5.5. Bamboo

Zur Qualitätssicherung gab es die Anforderung, das Projekt kontinuierlich auf Kompilierfehler und auf Fehler in den Testfällen zu prüfen. Dafür wurde Bamboo, ein Server für kontinuierliche Integration (CI) der Firma Atlassian, eingesetzt. Mit Hilfe von Bamboo werden die Maven-Projekte (s. Kap. 5.4) der Projektgruppe, die sich im Git Code Repositorium befinden, automatisch gebaut und die entwickelten Testfälle ausgeführt. Der Ablauf nach einem Commit, wie er auf dem Bamboo Server der Projektgruppe konfiguriert wurde, wird folgend kurz beschrieben.

Zunächst wird der neue Commit automatisch von dem Bamboo Server ausgecheckt. Daraufhin wird das Projekt (wenn keine Kompilierfehler bestehen) gebaut und alle Tests durchgeführt (ein Build wird durchgeführt). Dazu wird das Framework Maven zur Hilfe genommen. Falls Kompilierfehler oder Fehler bei den Tests auftreten, so schlägt der Build fehl. Ist dies der Fall, wird eine E-Mail an die als verantwortlich zugeordnete(n) Person(en) gesendet. Automatisch wird die Person, die den letzten Commit gemacht hat, als verantwortliche Person zugeordnet. Zusätzlich können verantwortliche Personen manuell von jedem Nutzer hinzugefügt oder entfernt werden. Nach drei fehlgeschlagenen Builds wird außerdem eine E-Mail an jedes Mitglied der Projektgruppe gesendet, damit auch die anderen Personen von dem Problem unterrichtet werden und diese gegebenenfalls lösen können. Dies soll dazu führen, dass Fehler frühzeitig erkannt und behoben werden und somit auch die Codequalität steigern.

Die bekannteste und meistgenutzte Alternative zu Bamboo ist Jenkins, weshalb diese beiden vor einer Auswahl etwas genauer untersucht wurden. Da Jenkins und Bamboo zwei sehr mächtige und in der Praxis erprobte CI-Server sind, wurden keine weiteren genauer untersucht.

[4] http://maven.apache.org/guides

Jenkins wird deutlich häufiger als das noch relativ junge Bamboo genutzt. Entsprechend existieren auch deutlich mehr Plug-ins für Jenkins als für Bamboo. Allerdings existieren auch für Bamboo mehr als einhundertfünfzig Plug-ins. Zudem sind alle von *Powder* benötigten Funktionen, wie das Auschecken aus einem Git Repositorium und das Anstoßen eines Maven Builds, bereits standardmäßig vorhanden.

Aufgrund der hohen Nutzerzahl gibt es für Jenkins viele Webseiten, Foren etc., die sich mit dem Thema beschäftigen. Allerdings bietet die Firma Atlassian eine qualitativ hochwertige und ausführliche Dokumentation für Bamboo. Keiner der beiden Server konnte sich hinsichtlich der Zwecke der Projektgruppe als deutlich besser geeignet herausstellen, jedoch erschien Bamboo nach erstmaliger Benutzung etwas simpler im Umgang als Jenkins. Zudem hätte Jenkins noch aufgesetzt werden müssen, während Bamboo zu dem Zeitpunkt der Entscheidung bereits aufgesetzt war. Aus diesen Gründen fiel die Entscheidung letzten Endes auf Bamboo.

5.6. Enterprise Architect

Enterprise Architect ist ein Softwaremodellierungswerkzeug von Sparx Systems, mit dem UML-Diagramme modelliert werden können.

Enterprise Architect wurde benutzt, da nur für dieses Anwendungsprogramm das SGAM-Toolbox-Plug-in verfügbar war und es sich als Modellierungswerkzeug im Smart Grid-Kontext etabliert hat [Neu14a]. Enterprise Architect unterstützt modellgetriebene Generierung, wodurch für ein SGAM eine Struktur vorgegeben wird, die eine konsistente Verwendung unterstützt.

Des weiteren unterstützt Enterprise Architect *CIM* (Abschnitt 5.11) und es existiert daher ein zusätzliches und nützliches, jedoch im weiteren Verlauf des Projektes nicht genutztes Plug-in UMLbaT. Diese hätte es erlaubt, aus einem *CIM*-konformen Modell einen OPC UA-Adressraum zu erzeugen. Für diese Aufgabe wurde stattdessen später der UaModeler verwendet (s. 5.12).

5.7. Rational Software Architect

Der Rational Software Architect von IBM[5] ist ein mächtiges Werkzeug, das neben der Modellierung von UML-Diagrammen komplexe Code- und Modelltransformationswerkzeuge (für Java und C++) anbietet . Zusätzlich gibt es Erweiterungen, die den Softwareentwicklungsprozess in vielfältiger Weise unterstützen[6]. Der Rational Software Architect wurde von der Projektgruppe für die Erstellung der UML-Diagramme für die Dokumentation verwendet.

Da die Projektgruppe lediglich für die Anfertigung der Dokumentation ein UML-Modellierungswerkzeug benötigte, bestanden keine umfangreichen Anforderungen an das Werkzeug. Die Hauptanforderung der Projektgruppe war, dass es sich unter einem Lizenztyp verwenden ließ, der die gleichzeitige Nutzung durch verschiedene Projektgruppenmitglieder ermöglichte. Weiterhin sollte das Werkzeug alle gängigen UML-Diagrammtypen unterstüzten und möglichst vielen Projektgruppenmitgliedern bekannt sein.

[5] http://www-03.ibm.com/software/products/en/rational-software-architect-family (letzter Zugriff am 29.03.16)

[6] http://www.ibm.com/support/knowledgecenter/SS8PJ7_9.0.0/com.ibm.rsa_base.nav.doc/topics/crootintro_rsa_base.html (letzter Zugriff am 29.03.16)

Zur Auswahl standen der Rational Software Architect und das Werkzeug Visual Paradigm[7], da diese schon von einigen Gruppenmitgliedern genutzt worden waren. Die Entscheidung fiel auf den Rational Software Architect. Einige Projektgruppenmitglieder hatten bereits mit beiden Werkzeugen Erfahrung gesammelt und konnten RSA empfehlen, da Visual Paradigm zwar ebenfalls alle für die Projektgruppe notwendigen Funktionen bereitstellt, aber der Rational Software Architect in einigen notwendigen Details einfacher zu nutzen ist.

5.8. Entwicklungsumgebung

Da alle Projektgruppenmitglieder bereits gute Erfahrungen mit IntellJ Idea und/oder Eclipse gesammelt hatten, wurden diese beiden Entwicklungsumgebungen in Betracht gezogen. Vor- und Nachteile ergaben sich lediglich durch minimale Unterschiede in der Bedienung, die aufgrund der Vorkenntnisse jedoch irrelevant waren. Da es auch durch den Einsatz des von Entwicklungsumgebungen unabhängigen Buildtools Maven (s. Kap. 5.4) kein Problem ist beide Entwicklungsumgebungen simultan zu nutzen, wurde die Wahl der Entwicklungsumgebung jedem Projektgruppenmitglied freigestellt.

Im späteren Verlauf des Projektes wurde das Plug-in *Saros* entdeckt, welches die Möglichkeit bietet, in Echtzeit kollaborativ über ein Netzwerk (sowohl über das Internet, als auch über ein LAN) an dem gleichen Projekt zu arbeiten. Dieses Plug-in wurde in Eclipse eingesetzt, da die Installation dort sehr einfach ist. Bei IntelliJ IDEA ist die Installation jedoch deutlich komplexer, weshalb diese nicht umgesetzt wurde. Stattdeseen wurde für IntelliJ IDEA das Plug-in *Floobits* verwendet, welches in etwa die gleiche Funktionalität zur Verfügung stellt wie *Saros*. Somit ergab sich keine Möglichkeit eines kollaborativen Arbeitens an dem Projekt mit unterschiedlichen Entwicklungsumgebungen. Ausschlaggebend für eine Umstellung auf eine der beiden Entwicklungsumgebungen war dies jedoch nicht, da das kollaborative Arbeiten an Projekt nicht als notwendig, sondern nur als optional erachtet wurde. Einige Mitglieder haben sich in Folge dessen beide Entwicklungsumgebungen installiert, wodurch eine kollaborative Arbeit auch mit anderen Mitgliedern in diesen Fällen ermöglicht wurde. Falls beide Entwicklungsumgebungen installiert und aktiv waren, so konnte durch automatische Hintergrundaktualisierung und bestehender Verbindung mittels einer der Plug-ins, auch zwischen den Nutzern verschiedenener Entwicklungsumgebungen kollaborativ gearbeitet werden. Dies konnte jedoch nur mit Verzögerung geschehen und wurde deshalb praktisch nicht eingesetzt.

5.9. Logging

Für *Powder* wurde ein eigener Loggingmechanismus entwickelt, welcher auf Log4j aufbaut. Dieser sollte eingesetzt werden, um das Debugging während der Entwicklung mit spezifischen Ausgaben zu vereinfachen. Damit der Endbenutzer die für ihn irrelevanten Ausgaben nicht sieht, gab es die Anforderung diese leicht deaktivieren zu können.

Log4j nutzt hierarchisch geordnete *Loglevel* für jeden Logeintrag:

TRACE < DEBUG < INFO < WARN < ERROR < FATAL

[7] `https://www.visual-paradigm.com/` (letzter Zugriff am 29.03.16)

Je nach Systemkonfiguration werden nur Logeinträge ab einem bestimmten Level in die Logdatei geschrieben. Liegt das aktuelle Loglevel beispielsweise auf WARN, so werden nur Logeinträge mit dem Loglevel WARN, ERROR und FATAL in die Logdatei eingetragen. Liegt das aktuelle Loglevel auf TRACE, so werden alle Logeinträge in die Logdatei eingetragen. Loglevel und weitere Konfigurationen können in der Datei log4j.properties angegeben werden. Ein Code-Ausschnitt, um mit Log4j einen Logeintrag mit dem Loglevel WARN zu erzeugen, wird im Code-Ausschnitt 2 beispielhaft gezeigt.

Code-Ausschnitt 2: Loggen mit Log4j

```
1  log4j.warn("Nachricht");
```

Damit diese Funktionalität um die Anforderungen von *Powder* erweitert werden, wurde ein eigener Loggingmechanismus (LVK) entwickelt, welcher auf Log4j aufbaut. Es sollte nicht, wie in Log4j üblich, nach Java-Klassen geloggt werden können, sondern nach Funktionalitäten. So können über mehrere Java-Klassen hinweg an zentraler Stelle die Loglevel für verschiedene Funktionalitäten eingestellt werden. Gleichzeitig kann eine Java-Klasse für mehrere Funktionalitäten mit unterschiedlichem Loglevel loggen. Um im LVK einen Logeintrag mit dem Loglevel WARN zu erzeugen, ist zusätzlich die Funktionalität anzugeben. Im Code-Ausschnitt 3 wird für die Funktionalität *database* geloggt. Der Logeintrag wird in der Klasse *LoggerTest* in der Methode *callerMethod* mit der Lognachricht *Message* und dem Loglevel *WARN* erstellt.

Code-Ausschnitt 3: Loggen mit LVK

```
1  public class LoggerTest {
2      public void callerMethod() {
3          LVK.database("Message", Level.WARN);
4      }
5  }
```

In der bereits erwähnten log4j.properties ist die Repräsentation des Logeintrags in der Logdatei konfigurierbar. Folgend die Konfiguration die für *Powder* gewählt wurde und ein beispielhafter Logeintrag.

Log Zeitpunkt	Log Level	Klasse	Methode	Zeile	Funktionalität	Log Nachricht
2016-03-02 11:57:26	**WARN**	**LoggerTest:**	**callerMethod:**	**2**	**- database:**	**Message**

Weitere Beispiele zu Log4j und die vollständige Dokumentation sind unter `http://logging.apache.org/log4j` zu finden.

Es wird zusätzlich zum Logging in der aufrufenden Konsole ebenfalls eine Logdatei angelegt, welche später und während der Ausführung eingesehen werden kann. Diese kann insbesondere genutzt werden, um während langwieriger Ausführungen, ungezwungen weiterzuarbeiten, ohne die Konsolenbindung aufrechtzuerhalten.

5.10. OPC UA

Für eine mögliche Anbindung von realen Anlagen an *Powder* ist es notwendig, ein passendes Kommunikationsprotokoll zu verwenden. Das Protokoll soll eine möglichst hohe Industrieakzeptanz besitzen und einfach zu integrieren sein. Daher sollte es auf einer höheren Abstraktionsebene angesiedelt und idealerweise bereits frameworkartig sein. Folgende wurden als mögliche Kandidaten in Erwägung gezogen:

- *JSON über HTTP*
- *IEC 61850*
- *Webservice*
- OPC Unified Architecture (OPC UA)

JSON über HTTP konnten früh als Protokollkandidaten ausgeschlossen werden, da eine sichere Integration zu aufwändig für das Projekt wäre.

IEC 61850 ist ein verbreiteter Kommunikationsstandard für Schutztechnik und Schaltanlagensteuerung. Er ist für diesen marktdienlichen Anwendungsfall somit eher ungeeignet [Sie16].

Web Services besitzen den Nachteil, dass eine sichere Verbindung ebenso zusätzlichen Aufwand erfordert und die Performance relativ gering ist [Usl16].

Schließlich fiel die Entscheidung auf OPC UA, da es bereits im Bereich der Anlagenkommunikation eingesetzt und, im Gegensatz zu Webservices, als künftige Schlüsseltechnologie im Bereich von Intelligenten Netzen betrachtet wird [USR+13, 209ff.].

OPC UA wird über eine öffentliche Spezifikation definiert. Dementsprechend existieren bereits viele quelloffene oder kommerzielle Implementierungen in vielen Programmiersprachen. Diese Plattformunabhängigkeit ermöglicht es unter anderem, dass ein eingebetteter Mikrocontroller mit komplexen Softwaresystemen kommunizieren kann [OPC16b]. Zudem stellt OPC UA verschiedene Methoden zur Authentifizierung und Verschlüsselung bereit, so dass eine sichere Kommunikation einfach integriert werden kann [OPC16a].

5.11. CIM

Das Common Information Model (CIM) ist ein Standard, der für den Daten- und Informationsaustausch in der Energiebranche von großer Bedeutung ist. Durch diesen Standard wird in Verbindung mit OPC UA die Kommunikation zwischen verschiedenen Anlagen, Steuerkomponenten und Systemen ermöglicht. CIM ist ein wichtiger Bestandteil der IEC 61970 Norm. Diese ist Teil eines von der *CIM user group* veröffentlichten Enterprise Architect Projekts[8], in dem viele Smart Grid relevanten Komponenten bereits modelliert sind. Eine ausführliche Beschreibung ist auch in Kapitel 8 des Seminarbandes zu finden (s. Anhang G).

[8] http://cimug.ucaiug.org/CIM%20Releases/iec61970cim15v33_iec61968cim11v13_iec62325cim01v07.zip

5.12. UaModeler

Der *UaModeler* bietet die Möglichkeit Informationsmodelle, wie beispielsweise ein CIM zu erstellen und dieses als XML zu exportieren. Die Software wurde ausgewählt, da sie in Verbindung mit Prosys OPC UA eingesetzt wird und zudem von der OPC Foundation angeboten wird. Somit kann die Kompatibilität sichergestellt werden.

Das Enterprise Architect - Plug-in *UML-BaT* konnte aufgrund von Kompatibilitätsproblemen mit dem Codegenerator, der für die Erzeugung von Java-Klassen benötigt wird, nicht genutzt werden. Weitere Informationen zu UML-BaT und UaModeler sind unter `http://www.umlbat.de/` und `https://www.unified-automation.com/de/produkte/entwicklerwerkzeuge/uamodeler.html` zu finden.

5.13. Weka

Für die Berechnung der Marktprognose bietet sich die Verwendung eines Machine Learning Frameworks an. Waikato Environment for Knowledge Analysis, kurz *Weka*, ist eine verbreitete Sammlung von Machine Learning und Data-Mining Algorithmen der University of Waikato in Neuseeland. Es steht unter der GNU General Public License. Die Wahl fiel auf dieses Framework, weil es in Java geschrieben ist und Zeitreihenanalysen unterstützt. Weka hat eine umfangreiche grafische Oberfläche, in der Datensätze geladen und historische Daten wie Vorhersagen visualisiert werden. Bei Weka liegt der Fokus in der Benutzung der grafischen Oberfläche.

Die Quelloffenheit ermöglicht den Zugriff auf Weka-Klassen und deren JavaDoc. Eine grobe Anleitung dazu ist auf der Weka-Seite [9] zu finden.

5.14. Gurobi

In Abschnitt 6.9.2 wird die lineare Optimierung als Lösungsmöglichkeit für die Produktportfoliooptimierung (PPO) oder die Gesamtoptimierung erläutert. Es existieren zahlreiche Tools, die für die Lösung eines linearen Optimierungsproblems verwendet werden können. Diese Tools werden teilweise bereits über Jahre auf bestmögliche Performance optimiert. Aus diesem Grund wurde auf eine eigene Implementierung verzichtet. Bei der Auswahl eines geeigneten Tools waren folgende Aspekte ausschlaggebend:

- Vorhandene Java-API
- Qualität der Dokumentation
- Performance
- Verfügbarkeit von Lizenzen
- Umsetzbarkeit des OR-Constraints

Unter dem OR-Constraint wird hier die Anforderung verstanden, dass die Größen der Pakete bei der PPO entweder in einem bestimmten Intervall liegt oder 0 ist. Diese Bedingung wird im Abschnitt

[9] `https://weka.wikispaces.com/Use+WEKA+in+your+Java+code`

zur linearen Optimierung im Kapitel zur Optimierung des Produktportfolios (6.9.2) aufgestellt. Das Constraint soll im Programmcode einfach umzusetzen sein.

Es wurden folgende Tools betrachtet:

- Open-Source-Software
  - GLOP
  - IP-Solve
  - CLP
  - GLPK
- Kommerzielle Software
  - Gurobi
  - CPLEX
  - Xpress

GLOP konnte schnell ausgeschlossen werden, da es kaum Dokumentation gibt und keine Java-API vorhanden ist. CLP besitzt ebenfalls keine Java-API. Aus Benchmarktests konnte ermittelt werden, dass die kommerziellen Optimierer eine deutlich bessere Performance aufweisen, als die Open-Source-Programme [Mit15]. Besonders GLPK und IP-Solve schnitten sehr schlecht ab und wurden aus diesem Grund aussortiert.

Alle drei kommerziellen Optimierer bieten akademische Lizenzen an und verfügen über Java-APIs mit ausführlicher Dokumentation. Außerdem ist es bei allen möglich, durch die Verwendung bestimmter Datentypen die OR-Constraints umzusetzen. Sie erfüllten damit alle Anforderungen und erschienen gleich gut geeignet. Daher wurden Codebeispiele und das Selbstausprobieren der Tools zu ausschlaggebenden Faktoren bei der Entscheidungsfindung.

Gurobi schnitt bei den meisten Benchmarktests am besten ab und konnte einfach und verständlich auf die im Projekt vorliegenden Optimierungsprobleme angewendet werden. Aus diesem Grund fiel die Wahl auf Gurobi.

5.15. Evaluation der Technologien

In diesem Abschnitt werden die in den vorhergegangenen Kapiteln vorgestellten Technologien evaluiert. Auf eine Evaluation der Technologien Java, MySQL, H2, Rational Software Architect und Gurobi wird jedoch verzichtet, da diese keine nennenswerten Probleme verursacht und Ihren Zweck erfüllt haben. Ob dieser durch den Einsatz anderer Technologien besser hätte erfüllt werden können, kann nicht beurteilt werden.

Hibernate Durch den Einsatz von Hibernate konnten Java-Objekte direkt in die Datenbank gespeichert werden, ohne die Notwendigkeit einer manuellen Konvertierung. Dieses ist für einfache Datenobjekte sehr einfach und hat viel Arbeit erspart. Für kompliziertere Datenobjekte kann dieses jedoch sehr komplex werden. Werden beispielsweise Listen von Objekten in einer Klasse benötigt,

ist dieses zwar Problemlos möglich, jedoch können die Datenstrukturen nicht so frei gewählt werden, wie gewohnt. So musste beispielsweise die Klasse `Collection` genutzt werden, anstatt der Klasse `List`, welche mehr Funktionalität bereitstellt.

Es wurde keine Lösung gefunden, um Daten der gleichen Klasse in verschiedene Tabellen zu speichern (beispielsweise um Wetterdaten aus verschiedenen Regionen voneinander zu unterscheiden). Stattdessen musste eine umständliche Lösung mit einer einzelnen Tabelle als Workaround verwendet werden.

Lombok Lombok konnte sehr einfach installiert werden und hat viel unnötigen Code eingespart. Dadurch wurde etwas Arbeit gespart und viele Klassen (insbesondere die meisten Modell-Klassen) wurden deutlich übersichtlicher.

Maven Maven hat das Erstellen von Jar-Dateien, welche auch alle benötigten Abhängigkeiten direkt beinhalten, deutlich vereinfacht. Dadurch konnten insbesondere die Skripte zur automatischen Generierung der Jar-Dateien (s. dazu auch 6.3) sehr einfach erstellt werden. Des Weiteren benötigt Bamboo ein Buildtool wie z. B. Maven, um das Projekt zu kompilieren und die Tests durchzuführen. Auch der Einsatz von zwei unterschiedlichen Entwicklungsumgebungen wäre ohne den Einsatz von Maven nicht möglich gewesen. Die vorgegebenen Ordnerstrukturen halfen außerdem dabei, die Konsistenz im Projekt zu erhalten.

Bamboo Bevor Bamboo genutzt wurde, kam es vor, dass fehlschlagende Tests erst nach längerer Zeit erkannt wurden. Dadurch wurde das Finden von Änderungen, die zu dem Problem führten, teilweise deutlich schwieriger. Zudem kam es dann häufig zu Folgefehlern, welche durch weitere Änderungen verursacht wurden. Durch die Nutzung von Bamboo konnten derartige Probleme frühzeitig erkannt und somit häufig leichter und ohne Folgefehler behoben werden.

SGAM Toolbox des Enterprise Architect Die Toolbox ermöglichte die Entwicklung von SGAM. Ohne sie wäre die Entwicklung deutlich schwieriger gewesen. Die Bedienung war allerdings nicht problemfrei:

- Teilweise automatisch generierte Verbindungen im Modell mussten wieder manuell versteckt werden.
- Funktionen sind nur über Umwege erreichbar.
- Ähnliche Funktionen sind verteilt statt an einer Stelle.
- Die Dokumentation einiger Funktionen war teilweise veraltet. Deshalb musste zunächst ausprobiert werden, mit welcher Funktion das gewünschte Ergebnis erzeugt werden konnte.
- Die Erzeugung neuer Elemente geschah immer auf unterschiedliche Art.
- Wenn die Toolbox selbst benötigt wurde, hat sie sich anschließend selbstständig wieder ausgeblendet. Ein Anpassen der Entwicklungsumgebung an die eigenen, notwendigen Funktionen schien nicht dauerhaft zu funktionieren.

Trotz allem konnte letztendlich das SGAM entwickelt werden. Daher bleibt nur, darauf zu warten, dass Anwendungsschwierigkeiten in kommenden Versionen berücksichtigt werden und die Dokumentation überarbeitet wird.

Entwicklungsumgebungen Dadurch, dass im Projekt zwei unterschiedliche Entwicklungsumgebungen (IntelliJ Idea und Eclipse) eingesetzt wurden, gab es Unterschiede bei Installationen von Plug-ins und einigen anderen Abläufen. Dadurch musste ggf. auch mehr Aufwand durch Erstellung von Anleitungen aufgebracht werden. Da sich die Anzahl derartiger Abläufe jedoch in Grenzen hielt und diese in der Regel recht einfach waren, hielt sich dieser Mehraufwand im Rahmen und eine Einigung auf eine einzelne Entwicklungsumgebung war nicht notwendig.

Es gab außerdem das Problem, dass die beiden Werkzeuge, die zum kollaborativen Arbeiten an *Powder* genutzt wurden, jeweils nur für eine der beiden Entwicklungsumgebungen eingesetzt werden konnte. Somit konnte nicht mit unterschiedlichen Entwicklungsumgebungen kollaborativ gearbeitet werden. Da das kollaborative Arbeiten jedoch nur als optional angesehen wurde, wurde dieser Umstand in Kauf genommen.

Abgesehen von den bereits aufgeführten Aspekten gab es keine Probleme durch den Einsatz von den unterschiedlichen Entwicklungsumgebungen.

LVK Durch den Einsatz von LVK konnten sehr einfach Log-Ausgaben erstellt werden. Es bietet u. a. die Möglichkeit, abhängig von der Funktionalität zu konfigurieren, welche Log-Ausgaben ausgegeben werden. Dadurch wird beispielsweise das Debuggen mit Hilfe von Log-Ausgaben vereinfacht, da die Informationen gefiltert werden können.

OPC UA Durch die Verwendung von OPC UA wurde die Verschlüsselung der Daten vereinfacht. Jedoch wurde es als schwierig empfunden, die Grundlagen von OPC UA zu verstehen, da keine schrittweisen Anleitungen oder einfach verständliche Beispiele zum Nachvollziehen existieren. Verwendete Plug-ins waren nicht auf allen OPC UA-Implementierungen anwendbar. Der verwendete Java-Codegenerator von Prosys erlaubt es in der verwendeten Evaluations-Version, nur maximal zehn Klassen zu modellieren, was zu Einschränkungen bei der Modellierung führte.

CIM Gute Beispiele vereinfachen das Verständnis und die Verwendung von CIM. Die CIM-Modelle ließen sich einfach als Klassendiagramme modellieren.

UaModeler Mithilfe des UaModelers konnten Definitionen und Konstrukte des OPC UA-Standards einfach zur Modellierung eigener Objekte verwendet werden. Allerdings konnten die erstellten CIM-Modelle nicht genutzt werden, sondern mussten eingeschränkt nachmodelliert werden, da der Codegenerator, wie zuvor bereits erläutert, auf zehn Klassen beschränkt war.

Weka Weka stellt viele Machine Learning Algorithmen zur Verfügung und verfügt über eine grafische Benutzeroberfläche, mit welcher die Algorithmen leicht angewendet werden können. Jedoch ist die Dokumentation ungenügend und es gibt es keine Programmierschnittstelle, wodurch das Verwenden der Algorithmen im Programmcode sehr erschwert wird.

6. Architektur

Ein Architekturentwurf stellt das Bindeglied zwischen der Analysephase und den Tests dar und beinhaltet damit die gesamte Softwarestruktur samt Entwurf und Implementierung im Detail. Der Architekturentwurf von *Powder* basiert auf den Erkenntnissen aus der Anforderungsanalyse. Entsprechend der Aufteilung in die verschiedenen HLUC (s. Unterabschnitt 4.4) erfolgt hier ebenfalls eine Strukturierung in unterschiedliche Module.

Auf die modulare Struktur samt Begründung der *Powder*-Architektur inklusive eines kurzen Überblicks der jeweiligen Module, sowie deren Abhängigkeiten untereinander, wird in Unterabschnitt 6.1 eingegangen.

Anschließend wird auf Alternativen für die Kommunikation der Module untereinander, sowie für die externe Kommunikation zu den Anlagen eingegangen. Außerdem wird die getroffene Auswahl begründet und die Implementierung erläutert. Als nächstes wird darauf eingegangen, wie innerhalb der Projektgruppe die Auslagerung und Ausführung des Projektes auf einen Server durchgeführt wurde.

Anschließend wird genauer auf jedes Hauptmodul eingegangen und jeweils Grundlagen, Implementierung und weitere nützliche Informationen detailliert vorgestellt.

6.1. Modulare Struktur

Wie bereits im Kapitel 6 erwähnt, besitzt *Powder* eine modulare Struktur: Aus jedem HLUC der Anforderungsanalyse ist ein Modul abgeleitet. Diese Module werden detailliert in den Folgenden Kapiteln beschrieben. Zuvor jedoch wird in diesem Kapitel zu jedem dieser Module eine kurze Übersicht gegeben. Neben den Modulen der HLUCs gibt es drei kleinere Module, auf die nur in diesem Abschnitt eingegangen wird.

Die modulare Struktur wurde durch die Projektgruppe gewählt, weil dadurch die Wiederverwendbarkeit erhöht werden konnte. So können spätere Projektgruppen oder andere Forschungsgruppen einen Nutzen aus einzelnen Modulen ziehen. Außerdem können diese leicht ausgetauscht werden. Ein weiterer Vorteil ist die Tatsache, dass sich die verschiedenen Module nicht auf demselben Rechner befinden müssen. Das hat zur Folge, dass z. B. rechenintensive Module ausgelagert werden können. Darüber hinaus kann gewährleistet werden, dass die Teilnehmer der Projektgruppe ohne Konflikte während der Programmierung gleichzeitig an den unterschiedlichen Modulen arbeiten können. Neben den bereits dargestellten Aspekten wird durch die Separierung die Komplexität der einzelnen Module geringer, als es bei einem einzelnen Projekt der Fall wäre. Außerdem wird dadurch die Kapselung der Funktionalitäten verbessert.

Da die einzelnen Module Abhängigkeiten untereinander besitzen, entstand ein Mehraufwand für die Kommunikation zwischen diesen, welcher jedoch aufgrund der Fülle an Vorteilen lohnenswert erschien. Abbildung 19 gibt eine Übersicht über die Module und die Datenflüsse zwischen diesen. Zu beachten ist, dass die Stammdaten und die Status der Anlagen von dem VPP-Modul zu den Anlagen gesendet werden. Dieses ist Notwendig, da in *Powder* keine tatsächlichen Anlagen eingesetzt werden und die simulierten Anlagen andernfalls nicht über diese Informationen verfügen. Sollten im produktiven Betrieb reale Anlagen vorhanden sein, so werden diesen keine Stammdaten und Status

vom VPP-Modul übergeben. Auf die Umsetzung der internen Kommunikation wird in Unterunterabschnitt 6.2.1 eingegangen.

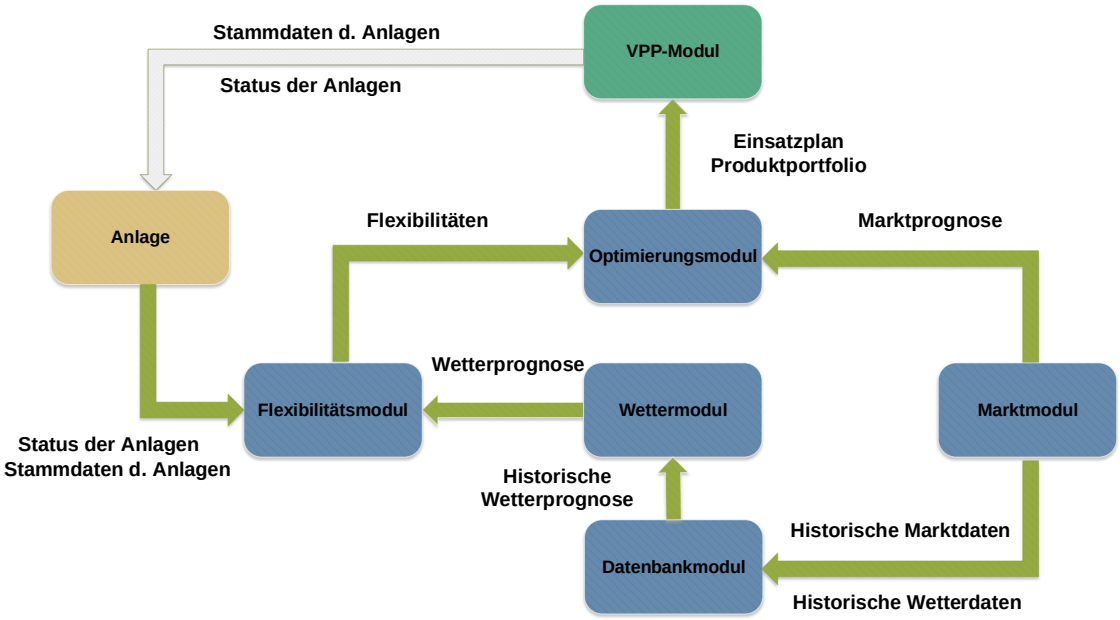


Abbildung 19: Modulstruktur mit Datenflüssen

Flexibilitätenmodul Das Flexibilitätenmodul bezieht Daten über die Anzahl und Art der vorhandenen Anlagen von dem VPP-Modul. Auf Basis dieser Informationen sowie der Wetterprognose instanziiert es für jede Anlage ein Modell mit den jeweiligen Parametern. Diese Parameter sind beispielsweise bei Blockheizkraftwerken u. a. Größe des Pufferspeichers, minimale Leistung und maximale Leistung. Die Modelle liefern, je nach Art der Anlage, beliebig viele (jedoch mindestens einen) gültige Fahrpläne und dessen Kosten zurück. Diese Fahrpläne geben an, wie viel Leistung eine Anlage über einen Zeitraum von 24 Stunden, unterteilt in 15-Minuten-Abschnitte, erzeugt. Diese Auflösung ist üblich für Anlagenfahrpläne. Diese Mengen gültiger Fahrpläne werden als *Flexibilitäten* bezeichnet und vom Optimierungsmodul benötigt.

Marktmodul Die Aufgabe des Marktmoduls ist, eine *Marktprognose* zu erzeugen und diese für das Optimierungsmodul bereitzustellen. Die Grundlage dafür sind die historischen Markt- und Wetterdaten. Die Marktprognose sagt zu einem gegebenen Datum den Strompreis auf dem EPEX Spot Markt pro Paket und Stunde vorher.

Es ist wahrscheinlich, dass Tage mit ähnlichem Wetter ähnliche Preise haben, da Windkraft- und Solaranlagen analog dazu bei ähnlichen Wind- und Sonnenständen vergleichbar viel Strom produzieren. Außerdem bewirkt eine vergleichbare Temperatur vermutlich eine vergleichbare Nachfrage. Deshalb wurde die Marktprognose im Prototyp mit einem selbst implementierten K-Nächste-Nachbarn Algorithmus erstellt. Spätere Versionen von *Powder* verwenden für die Marktprognose das Machine Learning Framework Weka. Dieses bietet einen besseren K-Nächste-Nachbarn Algorithmus, weshalb

die finale Version von *Powder* ausschließlich Algorithmen aus Bibliotheken für die Prognose verwendet.

Wettermodul Das Wettermodul wurde vom Prototyp zur Berechnung ähnlicher Wettertage benutzt. Spätere Versionen von *Powder* benötigen das Wettermodul nicht mehr, weil der Marktprognose-Algorithmus, welcher ähnliche Wettertage benötigte, durch maschinelles Lernen ersetzt wurde. Im praktischen Betrieb würde das Wettermodul die Wetterprognose für den nächsten Tag erhalten und diese aufbereitet für das Flexibilitätenmodul bereitstellen.

Optimierungsmodul Im Optimierungsmodul laufen die Informationen aus dem Marktmodul und dem Flexibilitätenmodul zusammen. Begrenzt durch die Flexibilitäten der Anlagen muss das Modul ein Produktportfolio, bestehend aus verschiedenen Produkten und Kapazitäten, erstellen, welches den Gewinn möglichst maximiert. Außerdem muss es für jede Anlage einen Fahrplan auswählen und anhand derer einen Einsatzplan erstellen, der den Betrieb aller berücksichtigten Anlagen umfasst. Diese beiden Erzeugnisse werden an das VPP-Modul übermittelt. Eine detailliertere Beschreibung der Optimierung erfolgt in Unterunterabschnitt 6.9.1.

VPP-Modul Das VPP-Modul soll die Funktionalität eines virtuellen Kraftwerks in beschränkter Form beinhalten. Als solches bildet es die Schnittstelle zu den Anlagen, indem es beispielsweise Stammdaten aus Textdateien ausliest und mit den Anlagen per OPC UA kommuniziert. Zudem koordiniert es die Ausführung der anderen Module und dient somit als Controller für den gesamten (Optimierungs-) Ablauf. Weiterhin beinhaltet das VPP-Modul auch die Schnittstelle zur GUI des Projektes und tauscht mit dieser Daten aus.

GUI Die grafische Benutzeroberfläche (GUI) befindet sich nicht in einem eigenen Modul, sondern ist als Paket ein Bestandteil des VPP-Moduls. Im GUI-Paket werden die Funktionen der grafischen Benutzerschnittstelle definiert und die Datenflüsse zu den Kernfunktionen von *Powder* koordiniert. Zudem wird durch diese die Berechnung gestartet.

Shared Das *shared*-Modul bietet Datenstrukturen, die von mehreren Modulen gleichzeitig benötigt werden. Hierzu zählen die Klassen für den Logger, Exceptions, RMI und eine Reihe von Datenmodellen, die zwischen den Modulen übertragen werden. Alle Klassen im shared-Modul, die mit RMI übertragen werden, müssen serialisierbar sein.

Registry-Modul Das Registry-Modul beinhaltet die Registry des zur internen Kommunikation genutzten *RMI* (mehr dazu in Unterunterabschnitt 6.2.1). Die *RMI*-Registry wurde in ein extra Modul ausgelagert, damit diese unabhängig von den anderen Modulen gestartet werden kann. Das Registry-Modul muss immer gestartet werden, bevor jegliche andere Aktionen, die mit RMI zusammenhängen, durchgeführt werden, da die anderen Module sich zunächst in der Registry registrieren müssen, bevor andere Module auf die *RMI*-Services zugreifen können.

Evaluationsmodul Während die bisher vorgestellten Module für die Funktionsweise des Gesamtsystems notwendig sind, dient das Evaluationsmodul der Evaluation der Optimierer. Es generiert verschiedene Szenarien, lässt jedes Szenario von jedem Optimierer durchführen und bereitet die Ergebnisse auf. Dabei bedient sich das Evaluationsmodul der anderen Module, um Marktdaten und Anlagenflexibilitäten für die ausgewählten Evaluationsszenarien erstellen zu lassen und an die Optimierer weiterzugeben.

6.2. Kommunikationsprotokolle

In diesem Kapitel wird zwischen der internen und externen Kommunikation unterschieden. Die Notwendigkeit interner Kommunikation ist bedingt durch die modulare Struktur der Architektur, die im vorherigen Kapitel erläutert wurde. Die interne Kommunikation findet zwischen den einzelnen, selbst entwickelten Modulen der Software statt, die im Verbund arbeiten. Im Abschnitt 6.2.1 werden verschiedene Technologien evaluiert und die Entscheidung für *RMI* begründet.

Als externe Kommunikation wird bei diesem Projekt die Kommunikation mit Dritten verstanden. Zur externen Kommunikation zählen einerseits der Zugriff auf historische Wetter- und Marktdaten und andererseits das Weiterleiten der Ergebnisse an die Anlagensteuerung und den Direktvermarkter sowie das Erhalten von Zustandsdaten der Anlagen. Da der Direktvermarkter bei der Umsetzung nicht näher betrachtet wird, bleibt für die externe Kommunikation nur der Datenaustausch mit den Anlagen und das Beziehen von Wetter- und Marktdaten. Der Zugriff auf Wetter- und Marktdaten wird in Abschnitt 6.2.2 beschrieben. Bei der Anlagenkommunikation kommt die in Abschnitt 5.10 vorgestellte Technologie *OPC UA* zum Einsatz. Die Umsetzung wird in Abschnitt 6.2.3 beschrieben.

6.2.1. Interne Kommunikation

Für die interne Kommunikation kommen verschiedene Technologien infrage, die im Folgenden beschrieben werden. Danach wird begründet, warum die Technologie (RMI) für *Powder* ausgewählt wurde.

ZeroMQ wurde von der Projektgruppe Ecob verwendet und bietet eine umfangreiche Bibliothek, mit der einzelne Komponenten kommunizieren können. Neben der Tatsache, dass ZeroMQ in C++ geschrieben ist, was die Verwendung etwas aufwändiger macht, ist der Umfang von ZeroMQ um einiges größer, als für dieses Projekt notwendig ist.

CIM-XML ist nur interessant, wenn das CIM als Informationsmodell verwendet wird. CIM-XML bietet die Möglichkeit, CIM-Objekte (und mehr) standardbasiert über HTTP zu kommunizieren. Es handelt sich dabei um eine Client/Server-Architektur.

Die *Remote Method Invocation (RMI)* ist eine Java-eigene Art des Remote Procedure Call (RPC). Sie wird dazu genutzt, entfernte Java-Methoden aufzurufen, als würden diese lokal vorliegen. Das heißt, eine Kodierung der zu übermittelnden Daten von Seiten des Entwicklers ist nicht notwendig. Als Rückgabe kann, wie in Java üblich, der entsprechende Datentyp der Funktion oder eine Exception gesendet werden. Im Folgenden wird zur Vereinfachung angenommen, dass es sich um eine Client-Server-Architektur handelt, bei der der Client bestimmte Funktionen auf dem Server aufrufen möchte. RMI setzt sich aus drei Komponenten zusammen: Dem *Remote Interface*, das die zur Verfügung

gestellten Funktionen definiert und auch der Client-Seite bekannt sein muss. Dem *Remote Object*, welches jenes Interface auf dem Server implementiert und der *Remote Reference*, die vom Server in die *RMI Registry* abgelegt wird und vom Client aus derselben gelesen werden kann, um damit auf die entfernten Methoden zugreifen zu können. Der schematische Ablauf wird in Abb. 20 dargestellt.

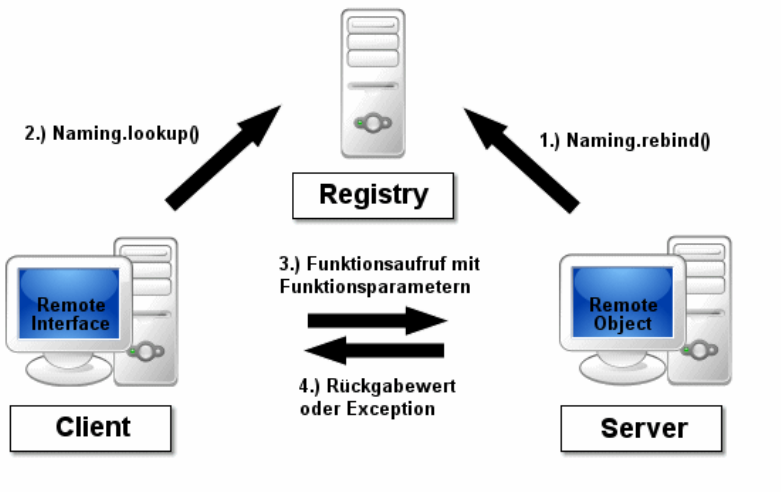


Abbildung 20: RMI Schema

Nach einer genaueren Betrachtung der verschiedenen Kommunikationstechnologien wurde RMI ausgewählt, da diese eine sehr einfache Benutzbarkeit gewährleistet. Weil für *Powder* außerdem keine zeitkritische oder durchgängige Kommunikation, z. B. in Form eines Streams, notwendig ist, sondern vielmehr Daten nur auf Anfrage geliefert werden sollen, bietet RMI hierfür ein passendes Konzept.

Implementierung Da die Architektur nicht nur aus klassischer Client-Server-Struktur besteht, sondern die Module stets zu mehreren anderen Modulen Abhängigkeiten haben (s. Abb. 19), muss dies bei der Implementierung berücksichtigt werden. Das führt dazu, dass jedes Modul sowohl Client- als auch Serverfunktionalität besitzen muss. Die Paketstruktur der für RMI entscheidenden Klassen wird in Abb. 21 visualisiert. Die einzelnen Klassen werden im Folgenden beschrieben.

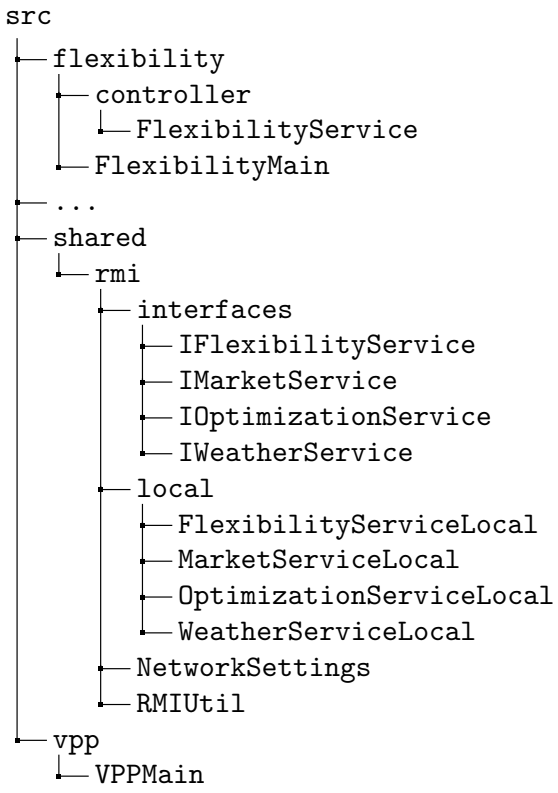


Abbildung 21: RMI Paketstruktur

Das `Registry`-Modul initialisiert die Registry, in welcher sich die anderen Module dann später registrieren. Die Remote Interfaces, die allen Modulen bekannt sein müssen, werden im `shared`-Paket abgelegt. Die Implementierung ihrer Funktionalität findet in den jeweiligen Modulen statt, z. B. im `FlexibilityService`. Dieser Service wird von der `FlexibilityMain` zur Registry hinzugefügt. Genutzt wird dazu die `rebindServer`-Methode von `RMIUtil`, die zugleich die entsprechenden Log-Ausgaben vornimmt.

Für den Zugriff auf die entfernten Methoden wird jeweils eine lokale Implementierung des Interfaces genutzt, welche die einzige Schnittstelle bilden soll (z. B. `FlexibilityServiceLocal`). Der Einsatz dieser Schnittstelle ist deshalb sinnvoll, weil so das Suchen über die Registry nur einmal vorgenommen werden muss und das erhaltene Objekt als Klassenvariable gespeichert werden kann. Zudem kann die `RemoteException`, die bei einem Verbindungsabbruch auftreten kann, so an einer Stelle behandelt werden. Z. B. indem entsprechende Logausgaben geschrieben werden. Die lokalen Implementierungen werden nur einmal benötigt, da sie sich nicht unterscheiden, sondern nur die im Interface definierten Methoden „weiterleiten“. Daher werden diese Schnittstellen ebenfalls im `shared`-Package gespeichert (unter `RMI/local`).

Alle notwendigen Parameter für die Netzwerk-Verbindung werden in den `NetworkSettings` statisch definiert, sodass diese von überall aus zugreifbar sind. Dort existiert für jedes Modul ein `Modul-NetworkSettings`-Objekt. Darin werden die vier Parameter: `modulLocation`, `Name`, `URL` und `Port` festgehalten. Da alle in der RMI-Registry eingetragenen Module an der gleichen Adresse liegen müssen, können entweder alle Module und die Registry auf dem Server ausgeführt werden oder alle Lokal

in der IDE oder als Jar-Datei. Der Name wird für den Zugriff über die RMI-Registry benötigt. URL und Port müssen je nach Server gesetzt werden. Über das Enum `ModulLocation` kann der Ort der Ausführung bestimmt werden:

Code-Ausschnitt 4: ModulLocation

```
1  public enum ModulLocation {
2    /** lokal in der Entwicklungsugebung ausführen */
3    IDE(true),
4    /** lokal getrennt von IDE ausführen (z.B. in jar) */
5    LOCAL(true),
6    /** aus einem Server ausführen (unter angegebener URL und Port) */
7    SERVER(false);
8
9    private final boolean isLocal;
10
11   private ModulLocation(boolean isLocal) {
12     this.isLocal = isLocal;
13   }
14
15   public boolean isLocal() {
16     return isLocal;
17   }
18 }
19 }
```

Damit individuell Anpassungen der `NetworkSettings` gemacht werden können, ohne dass diese das Git beeinflussen bzw. diese vom Git beeinflusst werden, wurde die Klasse `UserSpecificModulNetworkSettings` angelegt. In dieser vom Git ignorierten Klasse können individuelle Netwerk-Einstellungen konfiguriert werden. Wenn die Klasse vorhanden ist, werden die darin konfigurierten Netzwerk-Einstellungen automatisch bei Programmstart geladen. Falls die Klasse nicht vorhanden ist, werden die Standard-Einstellungen der `NetworkSettings` geladen.

6.2.2. Externe Kommunikation: Markt- und Wetterdaten

Weil eine direkte Verbindung zu Markt- und Wetterdatenprovidern wie dem Deutschen Wetterdienst nicht möglich war, wird der einmalige Import der Wetterdaten und der historischen Marktdaten und Gaspreise per Web Scraping gewählt. Dafür können die Daten mit Hilfe der mitgelieferten Python-Skripte von derEuropean Power Exchange Spotmarket (EPEX SPOT)-Website[10] abgerufen werden. Am Anfang der Python-Skripte kann der abzurufende Zeitraum eingestellt werden, für den die Werte ausgelesen werden sollen. Der Marktdaten-Scraper speichert die Block- und Stundenpreise jeweils in einer CSV-Datei für (`blocks.csv` und einer für `stunden.csv`). Jede Zeile dieser CSV-Dateien enthält das Datum, die zugehörigen Kosten in €/MWh und die Energiemenge in MWh. Diese Dateien können durch Ausführen des Java-Projektes `datenparser` in die Datenbank von *Powder* importiert

[10] https://www.epexspot.com/en/market-data/dayaheadauction/auction-table/

werden. Dadurch werden die Anforderungen S-03 und Anf. S-04 erfüllt. Für die Markt- und Wetterschnittstelle kann jedoch nicht sichergestellt werden, dass Anf. S-01 erfüllt ist.

Um neue Wetterdaten zu importieren, empfiehlt die Projektgruppe, den Datenparser an das Format der zu importierenden CSV-Datei anzupassen. Das Format des Datenmodells für die Datenbank wird in Unterunterabschnitt 6.5.3 beschrieben.

6.2.3. Externe Kommunikation: Anlagen

OPC UA wird als Protokoll zur Kommunikation mit realen Anlagen eingesetzt, damit Anf. S-02 erfüllt wird. Es hat sich gegenüber anderen Technologien durchgesetzt, da es plattformunabhängig ist, eine sichere Übertragung ermöglicht und als künftige Schlüsseltechnologie für Smart Grids gilt [OPC16a]. Neben der Übertragung von Informationen ist für OPC UA auch ein geeignetes Informationsmodell relevant. Für diese Aufgabe wird der für den Einsatz in Smart Grids bekannte CIM-Standard eingesetzt [Usl16, 209ff.]. Die zu entwickelnden CIM bauen auf IEC 61970 auf.

In OPC UA werden die kommunizierenden Komponenten in Client und Server eingeteilt. Zu einem OPC UA-Server können sich mehrere Clients verbinden, er stellt die Knotenadressen für Clients und hat die Fähigkeit, Daten zu veröffentlichen. Clients können von diesen Knoten Daten lesen, Daten schreiben oder Methoden aufrufen. Für die Standardisierung der Informationsobjekte wird CIM verwendet.

Als Hilfsmittel wird UML-BaT verwendet. Mit UML-BaT können Klassendiagramme in CIM-Struktur erstellt und in ein OPC UA lesbares XML exportiert werden. Die CIM-Struktur ähnelt der bereits bestehenden Klassenstruktur der BHKW- und PV-Anlagen im Flexibilitätenmodul. Für die CIM-Konformität werden die Einheitenklassen und der Fahrplan aus IEC 61970 benutzt. Außerdem muss die BHKW-Oberklasse von Identified Object erben. Das Informationsmodell für BHKW zeigt die Abbildung 22, das Informationsmodell für PV zeigt die Abbildung 23.

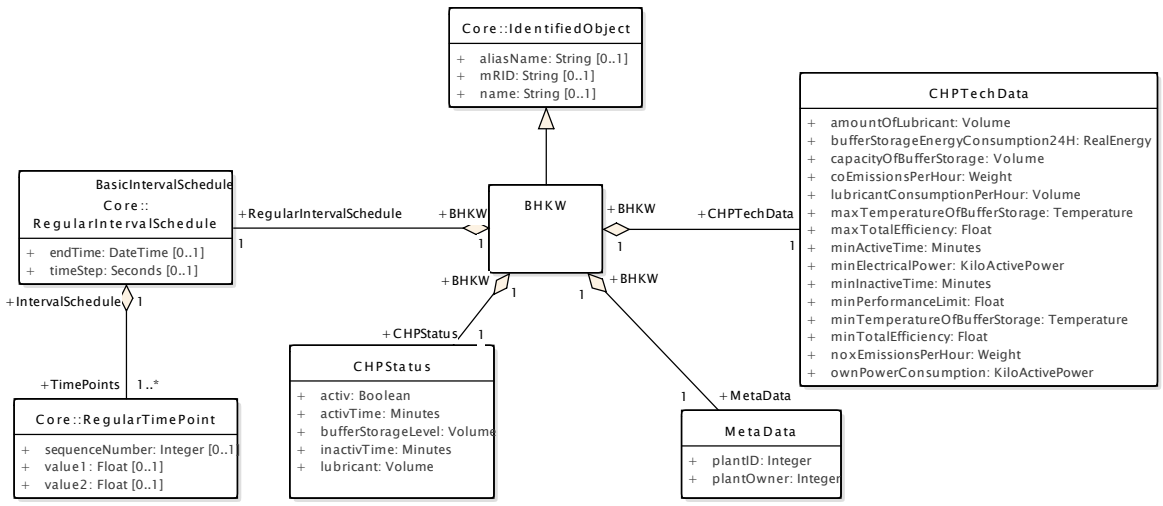


Abbildung 22: CIM Modell für BHKW

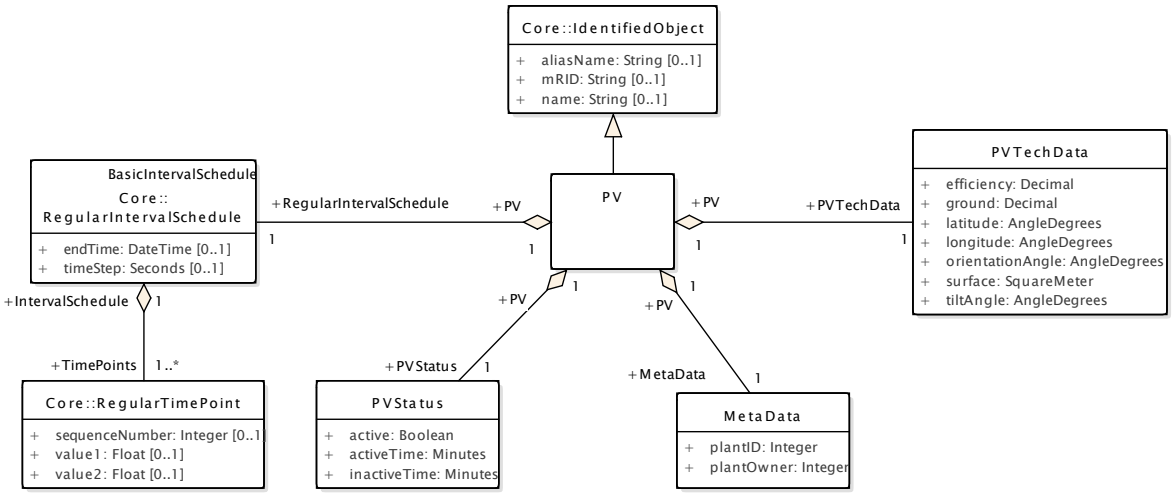


Abbildung 23: CIM Modell für PV-Anlagen

Implementierung Für das Projekt wird die Java Implementierung Prosys OPC UA[11] in der Evaluationsversion verwendet. Die Evaluationsversion ist vor allem dadurch eingeschränkt, dass eine Verbindung nur 120 Minuten bestehen darf und dann automatisch abgebrochen wird. Daher muss die notwendige Verbindung nur dann aufgebaut werden, wenn auch Daten übermittelt werden.

Damit aus dem CIM vor der Laufzeit die Java-Klassen erzeugt werden können, liefert Prosys einen Code Generator mit. Dieser ist durch die Evaluationsversion ebenfalls eingeschränkt und stoppt nach der Generierung von zehn Klassen. Dadurch werden Klassenabhängigkeiten nicht aufgelöst und unkompilierbarer Code generiert. Um die Klassenanzahl zu reduzieren, können die Einheitenklassen durch wenige primäre Datentypen und der Fahrplan durch ein 96 Felder großes double Array ersetzt werden. Dadurch kann CIM nicht korrekt umgesetzt werden und Anf. S-01 wird daher nur teilweise durch die Verwendung von OPC UA erfüllt.

Zudem herrscht eine generelle Inkompatibilität zwischen Code Generator und UML-BaT, sodass die vereinfachte CIM-Struktur in UaModeler nachmodelliert und als XML exportiert wurde.

Der OPC UA-Server wird von *Powder* implementiert. Die Anlagen müssen jeweils einen OPC UA-Client implementieren und ihre Daten in die Nodes des Servers eintragen. Dadurch liegen dem virtuellen Kraftwerk alle benötigten Daten vor. Außerdem wird es den Anlagen ermöglicht, Methoden aufzurufen, um sich beispielsweise zu registrieren.

Der OPC UA-Server ist im Flexibilitätenmodul eingebettet, der Client stellt ein eigenes Modul dar. Der Anlagen-Client dient nur dazu, die Kommunikation über OPC UA zu testen und enthält dementsprechend wenig Logik. Bei Ausführung des VPP-Moduls wird zunächst der Server gestartet und danach je nach Anlagenkonfiguration eine entsprechende Anzahl an Anlagen-Clients. Nach dem Verbindungsaufbau müssen sich die Anlagen beim virtuellen Kraftwerk registrieren. Der Server vergibt jeder Anlage eine ID, die die Anlagen auch als Rückgabewert erhalten. Mit der ID können Anlagen eindeutig auf ihre Knotenpunkte zugreifen. Im nächsten Schritt schreiben die Anlagen ihren Status und ihre Stammdaten in die ihnen zugewiesenen Knoten. Damit ist im Besonderen die Anforderung *Stammdaten mit Zustand der Anlage abfragen* (Flex-PUC1) erfüllt, auch wenn die Abfrage hier durch

[11] https://prosysopc.com/

passiven Informationszufluss ersetzt wird. Diese werden schließlich vom Server ausgelesen, der daraus die Anlagenobjekte erstellt, die für die nachfolgend stattfindende Optimierung benötigt werden (In diesem Fall Java-Objekte).

Nachdem die optimierten Fahrpläne erstellt wurden, werden diese von dem OPC UA-Server in die Knoten geschrieben. Abschließend lesen die Anlagen die Fahrpläne und geben sie in der Konsole aus, damit geprüft werden kann, ob die Kommunikation korrekt abläuft. Da lediglich Day-Ahead-Planung betrachtet wird, müssen die Anlagen nicht darauf hingewiesen werden, wenn die Fahrpläne geschrieben wurden. Stattdessen wird davon ausgegangen, dass die Fahrpläne zu einem festen Zeitpunkt vorliegen und ausgelesen werden können. Mit Hilfe eines Zeitstempels kann sichergestellt werden, dass der korrekte Fahrplan genutzt wird.

Dieser Kommunikationsablauf zwischen virtuellem Kraftwerk und Anlagen wird in Abbildung 24 dargestellt.

Die Verbindung wird gesichert, indem client- und serverseitig nur *SecurityMode.BASIC128RSA15_ SIGN_ ENCRYPT* erlaubt wird. Sobald sich ein unbekannter Client verbinden möchte, wird serverseitig ein Zertifikat erstellt. Dieses muss von dem VK-Betreiber akzeptiert werden, damit dem Client der Zugriff erlaubt wird.

Ungesichert bleibt jedoch der Zugriff auf fremde Knoten von bereits akzeptierten Clients. Eine weitere Authentifizierung für Knoten ist von OPC UA nicht vorgesehen. Für den Einsatz mit realen Anlagen kann davon ausgegangen werden, dass die Schnittstellen durch physikalische Isolation geschützt sind. Eine andere Lösung für dieses Problem kann eine Vertauschung von Client und Server sein, sodass Anlagen als Server und *Powder* als Client definiert würden. *Powder* könnte sich als Client zu mehreren Servern verbinden und erhielte die volle Sichtbarkeit über alle verfügbaren Knoten. Die Anlagen wären als Server nur auf die Sichtbarkeit ihrer eigenen Knoten eingeschränkt.

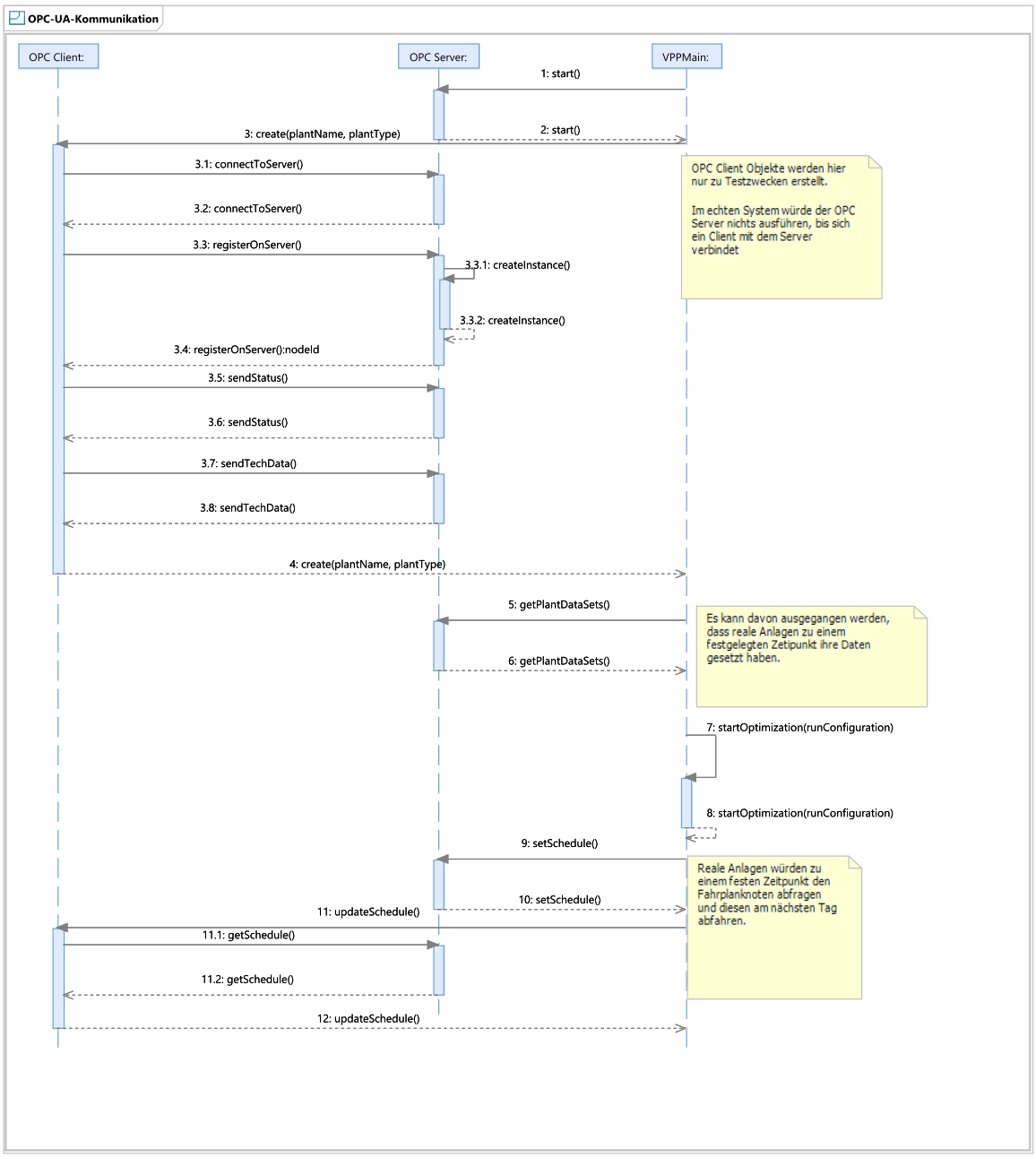


Abbildung 24: Nachrichtenaustausch zwischen OPC UA Client und Server

6.3. Auslagerung

Wie bereits im Kapitel Modulare Struktur erwähnt, ist die Möglichkeit, Module auf einen Server auszulagern, ein Grund für die modulare Aufteilung. Dies ist beispielsweise beim Ausführen von aufwendigen Prozessen, wie der Marktprognose oder der Evaluation, von Vorteil. Folgend wird beschrieben, wie die Serverauslagerung innerhalb des Projekts ermöglicht wurde.

Es müssen zunächst Jar-Dateien der einzelnen Module erstellt werden. Dies kann mit Maven automatisch durchgeführt werden. Zunächst mussten dafür jedoch die jeweiligen Maven-Projekte durch das *maven-assembly-Plug-in* entsprechend konfiguriert werden. Wichtig ist dabei nur die Angabe des Pfades zur Main-Klasse, folgend am Beispiel des Flexibility-Moduls illustriert:

Code-Ausschnitt 5: pom.xml

```
1 <mainClass>
2   de.pgvk.flexibility.FlexibilityMain
3 </mainClass>
```

Standardmäßig werden dadurch alle Abhängigkeiten mit in die Jar-Dateien eingebunden.

Damit die Prozesse ohne großen Aufwand wieder gestoppt werden können, wurde ein simpler Prozess implementiert, der dies durchführt. Dazu wird bei Programmstart eine temporäre Datei namens `Server.tmp` im Verzeichnis der ausgeführten Jar-Datei erstellt, falls diese noch nicht existiert. Es wird nun in regelmäßigen Abständen überprüft, ob die Datei noch vorhanden ist. Sobald dies nicht mehr der Fall ist, wird der Prozess beendet. Wenn die Jar-Dateien im gleichen Verzeichnis liegen, werden entsprechend alle Prozesse durch das entfernen der Datei beendet.

Folgend wird der konkrete Ablauf erläutert, welcher im Projekt durchgeführt wurde, um die Jar-Dateien auf einem Unix-basierten Server zu erstellen und das Projekt dort zu starten bzw. zu beenden, sowie die Funktionsweise der dazu benutzen Skripte. Voraussetzung ist, dass sowohl Maven als auch Git auf dem System installiert sind. Zunächst wurde das Git Repositorium an der gewünschten Stelle geklont. Nun muss die Klasse `UserSpecificModulNetworkSettings`, welche sich im Git-Unterordner `\other-projects\UserSpecificModulNetworkSettings` befindet, in das Shared Modul in das package `de.pgvk.shared.rmi` kopiert werden und entsprechend die Ports und Netzwerkadresse konfiguriert werden. Die `Location` muss außerdem für alle Module, die auf dem Server laufen sollen, auf `Server` gestellt werden.

In dem Unterordner `export-jar` des Git Repositoriums befinden sich alle benötigten Skripte, um die Jars automatisch zu bauen, sowie um die Prozesse zu starten und zu beenden. In der Datei `HowTo.txt` wird außerdem beschrieben, wie dieses umzusetzen ist. Durch Ausführen des Skriptes `createJarWithDependencieForAllModules.sh` werden alle Jars inklusive der Abhängigkeiten erstellt und in den Git-Unterordner `export-jar` kopiert. Dafür wird zunächst das Verzeichnis des Maven-Projekts `pg-vk-parent` gewechselt, welches das Überprojekt aller Module ist, und der Maven Befehl `mvn install -Dmaven.test.skip=true` ausgeführt. Dadurch wird das Projekt (inklusive der Unterprojekte) gebaut und die Abhängigkeiten auf dem lokalen Maven Repositorium installiert. Der Parameter `-Dmaven.test.skip=true` bewirkt, dass die Tests dabei nicht ausgeführt werden, da diese zu viel Zeit beanspruchen würden und die Testergebnisse nur während der Entwicklung von Bedeutung sind, jedoch nicht für die Auslagerung. Danach wechselt das Skript nacheinan-

der in die jeweiligen Ordner der Module und führt zunächst den Maven-Befehl `mvn clean compile assembly:single` aus, welcher die Jar-Datei erstellt und diese daraufhin in den `export-jar` Ordner kopiert. Nun kann durch Ausführen des Skripts `startAll.sh` bzw. `startEvaluation.sh` das Projekt bzw. die Evaluation gestartet werden. Das Skript `stop.sh` löscht die Datei `Server.tmp` und beendet somit die Prozesse.

6.4. VPP-Modul

Das VPP-Modul stellt, wie in Abschnitt 6.1 bereits angedeutet, das zentrale virtuelle Kraftwerksmodul von *Powder* dar. VPP steht dabei für *Virtual Power Plant*. Als zentrale Aufgaben des Moduls gelten die Aufgaben aus der Anforderungsanalyse, welche den Anforderungstabellen entnommen wurden. Es handelt sich insgesamt um sieben Hauptaufgaben.

- VK-Controller initialisieren
- Module starten
- Lokale Datenbank initialisieren
- Schnittstelle zum Wetterdatenprovider bereitstellen
- Schnittstelle zum Marktdatenprovider bereitstellen
- Schnittstelle zu den Anlagen bereitstellen
- Stammdaten der Anlagen initialisieren

Über die Anforderungen ist das VPP-Modul konzeptioniert und implementiert worden. Die genauere Beschreibung der Anforderungen kann dabei den Anforderungstabellen entnommen werden, auf die im jeweiligen Abschnitt referenziert wird. Im Folgenden wird der Entwurf des VPP-Moduls zu den einzelnen Anforderungen sowie deren Umsetzung dargestellt. Des Weiteren wird jede einzelne Anforderung in einem eigenen Abschnitt betrachtet. Anschließend wird auf weitere Implementierungen im VPP-Modul eingegangen.

6.4.1. Umsetzung der Anforderungen

VK-Controller initialisieren Für die erste Anforderung *VK-Controller initialisieren* (VPP-PUC1) wird im VPP-Modul eine Main-Methode benötigt, in der das Initialisieren des Controllers ausgeführt wird. Der Controller delegiert die weiteren Prozesse, die zur Optimierung des Produktportfolios ausgeführt werden. Es handelt sich daher auch um die zentrale Methode zum Starten des Projekts. Das VPP-Modul im Java-Projekt besitzt deshalb eine `VPPMain`-Klasse, welche die Eingangsmethode des Projekts beinhaltet. In dieser `main`-Methode wird zunächst die `VPPMain` initialisiert. Dabei wird das Startdatum `mStartDate` für die Optimierung und optional der Seed für die Zufallswerte eingelesen. Der feste Seed garantiert deterministische Zufallswerte, so dass bei Verwendung des gleichen Seeds dieselben Ergebnisse berechnet werden, selbst bei zufallsbasierten Verfahren, wie z. B. Heuristiken. Zudem wird in der `main`-Methode die `NetworkSettings.loadUserSpecificModulNetworkSettings`-Methode und die `RMI Start Configuration`-Klasse aufgerufen, sowie das Starten der externen Kommunikationsschnittstelle angestoßen. Durch diese Aufrufe in der `main`-Methode der Klasse VPPMain ist die Anforderung erfüllt. Die GUI kann die `VPPMain` ebenfalls starten. Dafür besitzt das VPP-Modul die Klasse `POWDERApp`, in der die GUI initialisiert und gestartet wird. Dieses ist zum einen für das Starten der Optimierung, also der VPP-Main, und zum anderen für

das Senden und Erhalten von Daten des jeweilig anderen Moduls nötig. Das VPP-Moul stellt im Projekt, wie bereits angedeutet, das virtuelle Kraftwerk dar. Somit ist auch die spezifische Anforderung E-01 umgesetzt.

Module starten Damit die Anforderung *Module starten* (VPP-PUC2) im VPP-Modul erfüllt werden kann, ist es notwendig, dass alle weiteren Module eine Schnittstelle, sowie Abhängigkeiten zum VPP-Modul besitzen. Zudem wird in diesem Zusammenhang ein interner Kommunikationsservice benötigt. Für das Starten der Module wird die bereits aufgezeigte Funktion `NetworkSettings.loadUserSpecificModulNetworkSettings` von der `VPPMain` verwendet. In dieser erfolgt die Registrierung der jeweils anderen Module. Die interne Kommunikation wird durch den Kommunikationsservice RMI übernommen. Die Entscheidung für und eine ausführliche Beschreibung der Funktionsweise von RMI kann dem Abschnitt 6.2.1 entnommen werden. Die Umsetzung des RMI-Services benötigt eine RMI-Konfiguration, welche durch die Klasse `RMIStartConfiguration` in der `VPPMain` umgesetzt wird. Die `RMIStartConfiguration` beinhaltet dabei alle nötigen Informationen, um den Server für RMI zu starten. Durch den Einsatz des internen Kommunikationsservices wurde neben der Anforderung *Module starten* auch die Anforderung nach einer internen Schnittstelle erfüllt.

Lokale Datenbank initialisieren Für die Anforderung *Lokale Datenbank initialisieren* (VPP-PUC3) ist vorgesehen, dass das VPP-Modul die Verbindung zur lokalen Datenbank aufbaut und jeweils eine Klasse zum Schreiben von Daten in die Datenbank sowie eine Klasse für das Auslesen von Daten bereitstellt. Da jedoch mehrere unterschiedliche Datenbankabfragen vorhanden sind, wurden die Datenbankoperationen in ein eigenes Modul ausgelagert, welches neben den Hauptmodulen besteht. Trotzdem wird die Anforderung erfüllt, da die Datenbankoperationen, durch das Starten des `db`-Moduls, indirekt in `VPPMain` initialisiert wird.

Schnittstelle zum Wetter- und Marktdatenprovider bereitstellen Da die Anforderungen *Schnittstelle zum Wetterdatenprovider bereitstellen* (VPP-PUC4) und *Schnittstelle zum Marktdatenprovider bereitstellen* (VPP-PUC5) ähnlich aufgebaut sind, ist eine fast äquivalente Umsetzung dieser beiden möglich. In einer Recherche über den Erhalt der Daten von dem jeweiligen Provider wurden zwei Möglichkeiten definiert, mit denen die Daten in das Projekt eingebunden werden können. Dabei kann die Abfrage der Daten über die Internetseiten bzw. Datenbanken der Provider erfolgen oder durch das vorherige Einspeichern der Daten in die eigene Datenbank und anschließender interner Datenabfrage. Mit Hilfe selbstgeschriebener Web-Scraper zum Abrufen der Daten wurde die zweite Möglichkeit umgesetzt. Die lokale Datenbank beinhaltet, nach Ausführung der Web-Scraper, alle Markt- und Wetterdaten, die vom Projekt benötigt werden. Daher werden die Anforderungen *Schnittstelle zum Wetterdatenprovider bereitstellen* und *Schnittstelle zum Marktdatenprovider bereitstellen* durch die erfüllte Anforderung *Lokale Datenbank initialisieren* mit erfüllt. Jedes Modul kann sich über das `db`-Modul und deren Methoden die gewünschten Daten aus der Datenbank auslesen. Damit sind die Anforderungen *Wetterprognose vom Wettermodul beziehen* (Flex-PUC2), *Wetterprognose abfragen* (Markt-PUC2) und die spezifische Anforderung *Das System muss eine Schnittstelle zur*

externen Datenbank der historischen Marktdaten besitzen (S-03) erfüllt. Auch das Einfügen neuer Daten kann über dieses Modul ausgeführt werden.

Schnittstelle zu den Anlagen bereitstellen Die Anforderung *Schnittstelle zu den Anlagen bereitstellen* (VPP-PUC7) beinhaltet die Anweisung, einen Kommunikationsservice bereitzustellen, mit welchem die Kommunikation zu den sich extern befindlichen Anlagen ausgeführt wird. Diese Kommunikation wird während der Durchführung der Optimierung benötigt, um wichtige Informationen, wie den Status der Anlage und deren Stammdaten, zu erhalten. Der ausgewählte Kommunikationsservice basiert auf dem Industriestandard OPC UA. Dieser besteht aus einem Server-Client-Modell, durch welches die Daten gesendet und empfangen werden. Details zum OPC UA-Standard lassen sich in Abschnitt 5.10 und Abschnitt 6.2 finden. Die Initialisierung des Servers und der Clients wird auch in der `VPPMain` ausgeführt. Dafür wird die Funktion `startOPCServerAndClients` ausgeführt. Damit der OPC UA-Service die zu verwendenden Anlagen registriert, werden in der `VPPMain` die Methoden `getPlantConfiguration` und `setPlantConfiguration` ausgeführt. Die Anlagenkonfiguration („PlantConfiguration“) wird dabei aus einer Textdatei *(.txt)* eingelesen, die zuvor während der Konfiguration in der GUI erstellt wird. Diese Datei enthält die Namen aller ausgewählten parametrisierten Anlagentypen und deren verwendete Anzahl. Diese Informationen werden über die Methode `setPlantConfiguration` für diejenigen Module und Klassen gesetzt, die diese benötigen. Durch das Starten des OPC UA Client-Server-Konstrukts im VPP-Modul und dem anschließenden möglichen Austausch von Daten zwischen dem Projekt und der Anlagen ist diese Anforderung erfüllt. Zudem wird so auch die spezifische Anforderung S-02 umgesetzt.

Stammdaten der Anlagen initialisieren und Datenblätter der Anlagen erstellen Die letzten für das VPP-Modul vorgesehene Anforderung sind *Stammdaten der Anlagen initialisieren* (VPP-PUC8) und *Datenblätter der Anlagen erstellen* (Stamm-PUC1). Hier werden neben der bereits vorhandenen „PlantConfiguration“ auch die Informationen über die Stammdaten benötigt. Diese werden, wie bereits beschrieben, über den externen Kommunikationsservice ausgelesen und sind vorher in die Anlage selbst gespeichert worden. Für die weitere Verwendung aller Daten, wie Anlagenanzahl, ausgewählte Anlagen, deren Stammdaten und Metadaten für die Nutzung durch die anderen Module, wird das Objekt `mTechDataManager` verwendet. Dieses beinhaltet eine Liste aller Anlagen (`mPlanDataSet`), sowie jeweils eine Hashmap für Blockheizkraftwerke (CHP) und Photovoltaik-Anlagen (PV-Anlagen). Diese Hashmaps besitzen zum einen den spezifischen Namen eines parametrisierten Anlagentyps, wie beispielsweise „CHP_LPG_0KW6“, sowie zum anderen die Anzahl, wie oft dieser Typ verwendet werden soll. Dazu werden in die Hashmaps die jeweiligen Stammdaten der Anlagen durch den Aufruf des `TechnicalDataLoader` ausgelesen und gespeichert. Der `TechnicalDataLoader` liest die Stammdaten aus *.txt*-Dateien, die durch eine Reader-Methode aus dem OPC UA-Service entnommen und dann durch eine Writer-Methode in die erwähnten *.txt*-Dateien geschrieben werden. Die Stammdaten werden in der Berechnung der Flexibilitäten verwendet, weshalb diese im Detail im Flexibilitätenmodul genauer erläutert werden. Weiterhin besitzt das Objekt die Anzahl der Fahrpläne, auf die jede Anlage begrenzt wurde, sowie das Datum der Berechnung. Der `mTechDataManager` ist in der Klasse `VPPService` angesiedelt. Diese Klasse wird wiederum in der

Klasse `VPPMain` aufgerufen. Durch die Verwendung der Klasse `TechnicalDataLoader` werden die Stammdaten initialisiert und können von jedem anderen Modul verwendet werden. Damit sind auch die letzten Anforderungen an das VPP-Moduls erfüllt. Des Weiteren ist es durch die Erstellung neuer *txt*-Dateien möglich, neue Anlagen zum virtuellen Kraftwerk hinzuzufügen, wodurch auch die spezifische Anforderung L-04 erfüllt wird. Zudem können die Stammdaten in den jeweiligen *txt*-Dateien so angepasst werden, dass die Werte von unterschiedlichen Herstellern eingesetzt werden können, womit auch die Anforderung L-12 erfüllt wird. Auch die Anforderung L-05 wurde umgesetzt, indem die *txt*-Dateien einfach gelöscht werden können.

Weitere Implementierungen Neben der Implementierung auf Basis der Anforderungen wurde noch eine Vielzahl an weiteren Klassen, Methoden und Funktionen im VPP-Modul implementiert. Dazu gehört auch die Klasse `HouseholdProfileMapper`, welche jeder BHKW-Anlage über die thermische Leistung einen der implementierten Haushaltstypen zuordnet. Dies ist in 6.4.2 näher erklärt. Auch hier wurde ein Datenleser implementiert, der alle Profildaten aus den Haushalts-*txt*-Dateien ausliest. Die Haushaltsprofile werden für die Berechnung des benötigten Wärmebedarfs der jeweiligen Verbraucher genutzt. Dies wird vom `HouseholdConsumptionCalculator` umgesetzt (s. Abschnitt 6.4.3).

Aus der bisherigen Beschreibung der Implementierung im VPP-Modul wird deutlich, dass eine Vielzahl von Datenlesern verwendet wird, die eine große Anzahl von Daten aus unterschiedlichen Dateien einlesen. Dies ist notwendig, da durch die zentrale Position des VPP-Moduls im Projekt die meisten Daten auch in den Klassen dieses Moduls benötigt werden oder von hier aus weitergeleitet werden können. Auch die Erstellung von neuen Dateien (zwecks Ausgabe von Ergebnissen) wird in diesem Modul häufig durchgeführt. Weiterhin werden die Ressourcen, wie Haushaltstypen, `TechData` und die `PlantConfiguration`, in diesem Modul gespeichert.

Als wichtigster Aufruf im VPP-Modul ist das Starten der Optimierung zu nennen. Dies wird durchgeführt, sobald alle dafür benötigten Daten vorhanden sind und die Infrastruktur (Kommunikation, Datenbankverbindung, usw.) aufgebaut wurde. Das Starten der Optimierung erfolgt in der Klasse `VPPMain` durch die Funktion `startOptimization`. Zum Start der Optimierung wird die Optimiererbeschreibung, welche Informationen über die zu verwendeten Optimierer enthält, benötigt. Dies ist notwendig, da zunächst mehrere Optimierer zur Verfügung standen, worauf eine Evaluation durchgeführt wurde, um den besten auszuwählen (6.9.5 und s. Abschnitt 6.10). Diese Beschreibung befindet sich in einer Textdatei *(.txt)* und wird über die Funktion `getOptimizerDescription` aufgerufen. Diese beinhaltet die Einstellungsparameter des Produktportfoliooptimierers (Innerer Optimierer) und des Einsatzplanoptimierers (Äußerer Optimierer). Dabei handelt es sich spezifische Werte des jeweiligen Optimierers, wie die Größe der Liste bei der Tabusuche. Diese Datei wird dann über einen `Loader` in die `startOptimization` eingelesen, damit die Optimierung ausgeführt werden kann. Das Ergebnis der Optimierung wird dann vom RMI-Service an die GUI übergeben.

Da alle verwendeten Kommunikationsschnittstellen auf Standards basieren, wird auch die spezifische Anforderung S-01 eingehalten.

6.4.2. Zuordnung eines Haushaltsprofils zu einem BHKW

Als spezifische Anforderungen an *Powder* wurde u. a. festgehalten, dass für die BHKW-Anlagen eine wärmegeführte Betriebsweise angenommen werden muss (L-13) und für den Wärmebedarf der Verbrauch eines typischen Einfamilienhaushalts zugrunde gelegt werden soll (L-14). Um allerdings eine größere Anzahl verschiedener parametrisierter Anlagentypen in *Powder* simulieren zu können, bedarf es verschiedener Haushaltstypen mit unterschiedlichem Wärmebedarf. Denn wird zu einem BHKW mit hoher thermischer Leistung ein Haushalt mit zu geringem Wärmeverbrauch zugeordnet, wird der Pufferspeicher des BHKW sehr schnell gefüllt, aber aufgrund des geringen Verbrauchs des Haushalts nur sehr langsam entleert werden. Dies hat zur Folge, dass das BHKW nicht so oft angeschaltet werden darf, damit die Maximaltemperatur des Pufferspeichers nicht überschritten wird (s. Abschnitt 6.8.6) und deshalb nur sehr wenig elektrische Leistung des BHKW produziert und zur Vermarktung genutzt werden kann. Folglich muss ein Haushaltstyp, der zu einem BHKW passen soll, einen thermischen Verbrauch besitzen, der ungefähr der thermischen Leistung des BHKW entspricht. Aus diesem Grund wurden in *Powder* neben dem Einfamilienhaushalt zehn weitere Haushaltstypen implementiert, wie beispielsweise Restaurant, Unterbringung oder Metallgewerbe. Für jeden Haushaltstyp wurden in Textdateien spezifische Verbrauchskonstanten hinterlegt, zum Beispiel der durchschnittliche Tagesverbrauch an Heizgas. Diese Daten wurden [FST06, BDE15] entnommen.

Um für ein BHKW einen passenden Haushaltstyp zu finden, wird zunächst für jeden der Haushaltstypen eine untere und eine obere Verbrauchsgrenze berechnet. Dazu wird der aus den jeweiligen Profildaten ausgelesene mittlere Wärmeverbrauch für einen Tag verwendet. Anschließend werden alle Haushaltstypen ermittelt, die zu diesem BHKW passen. Ein Haushaltstyp passt zu einem BHKW, wenn sich die thermische Leistung der Anlage innerhalb der Verbrauchsgrenzen befindet. Von diesen möglichen Haushaltstypen wird zufällig einer ausgewählt. Falls kein möglicher Typ gefunden wurde, wird der Haushaltstyp ausgewählt, dessen Verbrauch am ehesten der Anlagenleistung entspricht. Abschließend wird in beiden Fällen die Verbrauchskonstante des Haushalts an die thermische Leistung des BHKW angepasst, damit Leistung und Verbrauch noch besser zusammenspielen:

$$\text{Verbrauchskonstante} \;=\; \text{thermische Leistung des BHKW am Tag} \cdot \frac{2}{3} \tag{1}$$

6.4.3. Modellierung der Wärmeanforderungen eines Haushalts

Die Flexibilitäten der BHKW-Anlagen hängen stark von der Temperatur ihres jeweiligen Pufferspeichers ab (s. Abschnitt 6.8.6). Die Pufferspeichertemperatur wird von der BHKW-Leistung und vom Wärmeverbrauch des Haushalts beeinflusst. Aus diesem Grund muss die Wärmeanforderung des Haushalts in viertelstündlicher Auflösung vorliegen. Um den Tageswärmebedarf des Haushalts zu berechnen, wurde das Standardlastprofilverfahren (SLP-Verfahren) des Bundesverbandes der Energie- und Wasserwirtschaft verwendet [BDE15]. Mit Hilfe des SLP-Verfahrens kann der geschätzte Gasverbrauch eines Haushalts tagesgenau berechnet werden, wobei die prognostizierte Temperatur des jeweiligen Tages einbezogen wird. Anschließend wurde der Heizgasverbrauch in die reale Wärmeanforderung des Haushalts umgerechnet. Mit der Umsetzung der Wärmeanforderung werden auch die spezifischen Anforderungen L-13 und L-14 eingehalten.

SLP-Verfahren Sei D der Tag, für den die Tagesverbrauchsmenge an Heizgas berechnet werden soll. Hierfür wird folgende Formel herangezogen [BDE15, S. 41 ff]:

$$Q_D = KW \cdot h(\vartheta) \cdot F_{WT} \tag{2}$$

Q_D prognostizierter Heizgasverbrauch für den Tag D [kWh]

KW Kundenwert [kWh] (mittlerer Tagesverbrauch pro Tag; aus historischen Daten individuell für den Haushalt ermittelt)

ϑ Allokationstemperatur [°C]

$$\vartheta = \frac{T_D + 0,5 \cdot T_{D-1} + 0,25 \cdot T_{D-2} + 0,125 \cdot T_{D-3}}{1 + 0,5 + 0,25 + 0,125} \tag{3}$$

mit T_D prognostizierte Temperatur für Tag D, T_{D-1} prognostizierte Temperatur für den Vortag $D-1$ usw..

h Profilfunktion

$$h(\vartheta) = \left[\left(\frac{A}{1 + \left(\frac{B}{\vartheta - \vartheta_0} \right)^C} + D \right) \right] + max\{m_H \cdot \vartheta + b_H, m_W \cdot \vartheta + b_W\} \tag{4}$$

wobei A, B, C, D, ϑ_0, m_H, m_W, b_H, b_W Konstanten sind, die dem Standardlastprofil des Haushalts entnommen werden können. Diese sind aus den Quellen [BDE07] und [BDE15] auslesbar.

F_{WT} Wochentagfaktor (Einfluss des Wochentags auf den Verbrauch)
Konstante, die ebenfalls dem Standardlastprofil entnommen werden kann.

Berechnung des realen Wärmebedarfs Zunächst wird der Tagesverbrauch auf die einzelnen Stunden umgerechnet. Dafür wurden Standardwerte aus den Quellen [BDE07] und [BDE15] verwendet, die für jede Stunde des Tages angeben, welcher Anteil des gesamten Tagesverbrauchs normalerweise abgerufen wird. Anschließend wurden die Werte für jede Stunde nochmals zu gleichen Teilen auf die Viertelstunden aufgeteilt, da dies für die Bestimmung der Anlagenflexibilitäten notwendig ist.

Für die Bestimmung des realen Wärmebedarfs des Haushalts muss der im SLP-Verfahren berechnete, auf die Viertelstunden aufgeteilte Heizgasverbrauch, umgerechnet werden. Der Haushalt verbraucht nicht so viel Energie, wie im verbrauchten Heizgas enthalten ist, da der Wirkungsgrad jedes normalen Heizkessels unter 100 Prozent liegt. Die reale Wärmeanforderung wurde berechnet, indem für jede Viertelstunde der Verbrauch mit dem Wirkungsgrad des Heizkessels multipliziert wurde. Es wurde der Wirkungsgrad eines Niedertemperaturkessels verwendet. Dieser liegt nach [Hes] bei 93 Prozent.

6.5. Datenbankmodul

Das Datenbankmodul ermöglicht den Zugriff auf die MySQL-Datenbank via Hibernate. Folgend wird die Wahl des Datenbankmodells begründet. Danach wird auf die Implementierung und das Datenmodell anhand der für Hibernate benötigten Java-Klassen eingegangen.

6.5.1. Datenbankmodell

Die Projektgruppe hat sich für eine relationale Datenbank entschieden, weil sie mit der Einrichtung und Benutzung relationaler Datenbanken bereits vertraut war. Die Performance verschiedener Datenbankmodelle war bei der Entscheidung irrelevant, weil die Objekte im Java-Code zwischengespeichert werden sollten. Zu speichernde Daten waren beispielsweise historische Markt- und Wetterdaten. Diese Daten lagen in einer einheitlichen Struktur (einer geordneten Tabelle), wie beispielsweise einer *csv*-Datei, vor. Aus diesem Grund war es nicht notwendig eine NoSQL-Datenbank zu verwenden, welche das Speichern von unstrukturierten Daten ermöglicht. Ein weiterer Grund für die Verwendung einer relationalen Datenbank war die Nutzung von Hibernate. Hibernate ist zwar auch für NoSQL-Datenbanken verfügbar, die Nutzung von Hibernate für relationale Datenbanken war jedoch bereits mehreren Projektgruppenmitgliedern bekannt.

6.5.2. Implementierung

In der Datei `hibernate.cfg.xml`, welche sich im *resources* package befindet, wird Hibernate konfiguriert. Dort müssen Informationen wie Adresse und Zugangsdaten der Datenbank angegeben werden, aber auch eine Liste der voll-referenzierten Klassen, die als Modell für Datenbankdaten verwendet werden sollen. Jede Klasse, die in der Datenbank gespeichert werden soll, muss dabei mit `@Entity` annotiert werden und jedes Attribute mit `@Column`.

In der Klasse `DefaultDatabaseService` wird eine Hibernate `Session` zur Verfügung gestellt, welche den Zugriff auf die Datenbank ermöglicht.

Im folgenden Kapitel wird nun erläutert welche Klassen angelegt worden sind, um die benötigten Datenbankdaten zu modellieren.

6.5.3. Datenmodell

In diesem Abschnitt werden die Datenobjekte modelliert, die innerhalb der Software von den Modulen verwendet werden. Die Module *Wetter* (s. Abschnitt 6.6) und *Markt* (s. Abschnitt 6.7) benötigen jeweils eigene Datenobjekte, die in der Regel an andere Module zur Weiterverarbeitung gesendet werden.

Das Wettermodul hat die Aufgabe, Wetterdaten bereitzustellen. Es wird also ein Datenobjekt benötigt, welches diese Wetterdaten repräsentiert. Beim Entwurf des Datenobjektes muss berücksichtigt werden, welche charakteristischen Eigenschaften ein Wetterdatum ausmachen. Außerdem ist die zeitliche Auflösung festzulegen. Einerseits ändert sich das Wetter kontinuierlich, andererseits sind diese Änderungen meist marginal. Für die anschließende Berechnung der Marktpreise genügt eine stündliche Auflösung der Wetterdaten. Aus diesem Grund wurde eine Wetterstunde als zeitlich am kleinsten

aufgelöstes Wetterdatum modelliert, welches in Java durch die Klasse `WeatherHour` implementiert wurde. Diese `WeatherHour` muss Informationen über Temperatur, Sonnenstrahlung, Bewölkungsgrad und Windstärke, jeweils als Durchschnittswert, enthalten. Weitere Attribute, wie Niederschlag oder Luftdruck, sind nicht relevant genug für den Systemkontext. Zusätzlich wird gespeichert, welche Stunde des Tages ein `WeatherHour`-Objekt repräsentiert. Dabei beinhaltet z.B. die Stunde null das Zeitintervall von 00:00 Uhr (einschließlich) bis 01:00 Uhr (ausschließlich).

Für einen gesamten Tag werden vierundzwanzig `WeatherHour`-Objekte benötigt. Diese werden, wie in Abbildung 25 dargestellt, in einem zusätzlichen Datenobjekt, der `WeatherEntity` (*dt.* Wetterentität), gebündelt. In dieser enthalten ist auch das Datum des durch die Wetterstunden beschriebenen Tages. Des Weiteren gibt es das Enum `WeatherType` (s. Code-Ausschnitt 6), welches die Unterscheidung zwischen geographischen Gebieten, wie z. B. dem Wetter für Oldenburg, und Gesamtdeutschland ermöglicht. Zusätzlich wird dadurch aber auch die Unterscheidung zwischen historischen Wettervorhersagen und historischen Wetterdaten ermöglicht.

Code-Ausschnitt 6: WeatherType

```
1 public enum WeatherType {
2   UNKNOWN, OLDENBURG_PREDICTION, GERMANY_PREDICTION, OLDENBURG, GERMANY;
3 }
```

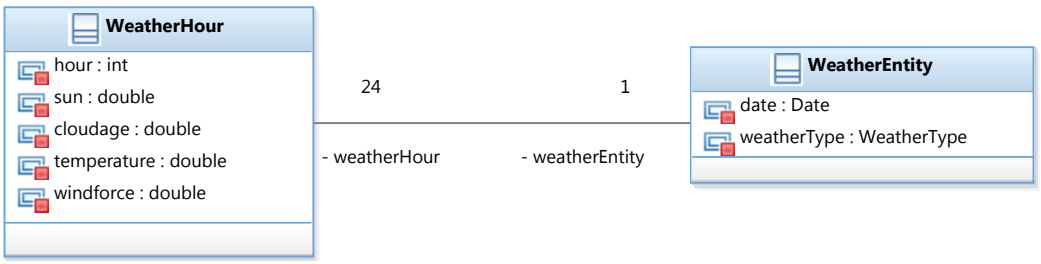


Abbildung 25: `WeatherEntity` und `WeatherHour`

Das Marktmodul benötigt Marktdaten, um aus diesen eine Marktprognose zu erstellen. Dazu werden zusätzlich die Wetterdaten benötigt, die wie oben beschrieben in stündlicher Auflösung vorliegen. Analog zur `WeatherEntity` wird eine `MarketEntity` (*dt.* Marktentität) benötigt, die einzelne `HourPrices` (*dt.* Stundenpreise) enthält. Ein `HourPrice`-Objekt muss sowohl die zu beschreibende Stunde, als auch den durchschnittlichen Strompreis des Zeitintervalls enthalten.

6.6. Wettermodul

Im Folgenden wird die Funktionsweise des Wettermoduls zum Zeitpunkt des Prototyps erläutert. Es hat auf Grundlage der historischen Daten für einen gegebenen Tag bzw. ein gegebenes Wetterprofil (in der Regel die Wetterprognose) die Tage ermitteln, die diesem am ähnlichsten sind. Diese wurden vom Marktmodul für die Marktprognose mithilfe eines K-Nächste-Nachbarn Algorithmus verwendet. Die Ähnlichkeit kann durch die akkumulierte Abweichung von bestimmten relevanten Faktoren, wie der Sonneneinstrahlung oder Windstärke, berechnet werden. Allerdings wird das Wettermodul in der aktuellen Version nicht mehr benötigt, da die Berechnung der ähnlichsten Wettertage in der Marktprognose mit Waikato Environment for Knowledge Analysis stattfindet.

6.6.1. Umsetzung der Anforderungen

Die Anforderungen *Historische Wetterdaten beziehen* (Wetter-PUC1) und *Wetterprognose beziehen* (Wetter-PUC3) benötigen eine Anbindung an einen externen Wetterdatenprovider wie z. B. den Deutschen Wetterdienst. Leider war es nicht möglich, einen Zugang zu einem Wetterdatenprovider zu bekommen. Aus diesem Grund wurden diese Anforderungen verworfen. Stattdessen wurde der Projektgruppe ein Datensatz mit historischen Wetterdaten und Wetterprognosen für das Jahr 2012 zur Verfügung gestellt. Weil die historischen Daten zur Verfügung stehen und *Powder* nicht entwickelt wurde, um im Produktivbetrieb genutzt zu werden, ist es nicht mehr nötig, aktuelle Daten zu speichern. Damit werden die Anforderungen *Historische Wetterdaten speichern* (Wetter-PUC2) und *Wetterprognose speichern* (Wetter-PUC4) obsolet.

Die Anforderung *Einzelne Wetterprognose abrufen* (Wetter-PUC5) wurde umgesetzt, indem die Wetterdaten per Hibernate-Verbindung aus der Datenbank geladen und bereitgestellt werden. Damit ist auch die spezifische Anforderung *Das System muss eine Schnittstelle zur externen Datenbank der historischen Wetterdaten besitzen* S-04 umgesetzt.

Die Anforderungen *Datenfilter anwenden* (Wetter-PUC7) sollte Wetterdaten auf Datenbankebene filtern, damit weniger Daten abgerufen werden müssen, weil dieses die Zugriffsgeschwindkeit verbessert. Stattdessen werden alle Wetterdaten einmalig abgerufen und intern zwischengespeichert. Dadurch muss der relativ langsame Abruf der Daten aus der Datenbank nur einmalig durchgeführt werden. Anschließend werden alle `WeatherEntity`-Datenobjekte nach ähnlichen Tagen zum betrachteten Tag gefiltert. Der Filter ist also nicht wie in der Anforderung beschrieben auf Datenbankebene, sondern auf Programmebene. Damit sind diese und die Anforderung *Datenfilter erstellen* (Wetter-PUC6) zwar verworfen, das Ziel, ähnliche Tage zu finden, ist aber erfüllt.

6.6.2. Implementierung

Dieser Abschnitt stellt die Implementierung der Berechnung ähnlichster Wettertage vor, die im Prototypen verwendet wurde. Wie bereits oben erwähnt, wurde dies stattdessen vom Marktmodul übernommen.

Die Methode `getSimilarDates` in der Klasse `WeatherData` gibt anhand eines Datums eine bestimmte Anzahl (n) an Daten zurück. Die zurückgegebenen Daten sind jene, die (in Bezug auf das Wetter) dem übergebenen Datum am ähnlichsten sind. Dafür wird mit Hilfe der `compareNot-`

`Weighted`-Methode (ebenfalls in der `WeatherData`-Klasse) das Wetter aller historischen Wetterdaten mit dem Wetter des übergebenen Datums verglichen und bestimmt, wie ähnlich sich diese sind. Wie genau die Ähnlichkeit bestimmt wird, wird im nächsten Absatz erläutert. Es werden nun die n Daten mit dem besten Ähnlichkeitswert zurückgegeben.

Die Methode `compareNotWeighted` vergleicht zwei `WeatherEntity`-Objekte, welche die Wetterdaten repräsentieren. Anhand der `WeatherAttribute`, wie z. B. der Temperatur, wird für jede der 24 Stunden die Abweichung, also z. B. die Differenz der Temperatur, berechnet. Danach wird der Durchschnitt dieser 24 Abweichungen ermittelt.

Dieser Durchschnitt wird dann auf einen Wert zwischen 0 und 1 normalisiert. Der Durchschnitt der jeweils normalisierten Werte aller Attribute bildet dann den Ähnlichkeitswert der zwei `WeatherEntity`-Objekte: Je geringer er ist, desto ähnlicher sind sich die beiden Objekte.

Teilweise existieren nicht für jede Stunde des Tages Messdaten, sondern nur für jede dritte Stunde. Entsprechend auftretende Lücken werden mittels Imputation kompensiert. Dafür gibt es diverse Ansätze. In *Powder* wurde ein Ansatz implementiert, der folgende Funktionsweise aufweist: An einem Tag wurden die Stunden null und drei gemessen. Die dazwischen liegenden Stunden eins und zwei fehlen. Nun wird für den Wert von Stunde eins $\frac{2}{3} \cdot$ Wetter von Stunde null $+ \frac{1}{3} \cdot$ Wetter von Stunde drei zurückgegeben.

Komplexere Ansätze, wie z. B. Regressionsverfahren, wurden nicht umgesetzt, da sich das Wetter innerhalb von drei Stunden nicht drastisch ändert und der Fokus von *Powder* nicht auf den Wetterdaten liegt.

6.7. Marktmodul

Um die optimalen Produkte für das Anlagenportfolio auszuwählen, ist es von großer Bedeutung, eine gute Prognose für die Preise der einzelnen Produkte des folgenden Tages zu erhalten. Denn je besser diese Prognose ist, desto genauer kann der Gewinn für jedes einzelne Produktportfolio berechnet werden, was wiederum die Auswahl des optimalen Produktportfolios verbessert. Die Aufgabe der Projektgruppe besteht deshalb auch darin, ein Prognosesystem für die Preise der einzelnen Produkte zu entwickeln.

Das Marktmodul hat die Aufgabe, Vorhersagen von Marktpreisen für ein angefordertes Datum zu erstellen und als Service bereitzustellen.

Nach der Recherche zu softwaregestützten Prognosen fiel das Hauptaugenmerk auf das maschinelle Lernen. Beim maschinellen Lernen wird aus Daten ein Modell generiert [Kel12]. Dieses Modell wird dann zur Prognose verwendet. Weil die Preise für ein bestimmtes Datum prognostiziert werden, kann aus den Preisen der vorhergegangenen Wochen und Monate gelernt werden. Es handelt sich also um eine Time Series Prediction, welche sehr gut mit Techniken des maschinellen Lernens umgesetzt werden kann.

Möglichkeiten der Marktprognose mit Machine Learning Maschinelles Lernen lässt sich in drei große Bereiche unterscheiden: Nicht überwachtes Lernen, überwachtes Lernen und bestärkendes Lernen.

Beim *nicht überwachten Lernen (Unsupervised Learning)* wird allein auf den Eingabewerten ohne bekannte Zielwerte gelernt. Die Eingabewerte werden dabei durch Segmentierung oder Komprimierung analysiert oder klassifiziert. Mögliche Algorithmen sind SVM (Support Vector Machine), k-NN (K-nearest neighbors), Regression, Relief-Algorithmus, Hauptkomponentenanalyse, Bayesian Networks und Clustering [Kel12].

Beim *überwachten Lernen* werden dem System Datenpaare übergeben, die aus einer Eingabe und deren Ergebnis bestehen. Aus den Eingabedaten versucht das System, eine Funktion abzuleiten. Mögliche Algorithmen für das überwachte Lernen sind Künstliche Neuronale Netze, Relief-Algorithmus und Regression (insb. Support Vector Regession [Kel12]).

Auch beim *bestärkenden Lernen (Reinforcement Learning)* besteht die Eingabe aus Datenpaaren mit Eingabe und deren Ergebnis. Der Fokus liegt aber auf dem Belohnen bzw. Bestrafen des Systems bei einer erfolgreichen oder nicht erfolgreichen Vorhersage. Oft beobachtet ein Agent die Effekte seines Handelns auf die Außenwelt. Er lernt aus den dabei entstehenden Erfolgen und Misserfolgen und passt sein Verhalten entsprechend an. Dieses Lernverfahren kommt vor allem in der Robotik zum Einsatz [Kel12].

Bei der Marktprognose liegen die historischen Daten (z. B. Datum, Temperatur) als *Merkmale* („x-Werte“) vor. Zusätzlich liegen die Preise als *Label* („y-Wert“) vor. Die Kombination aus Merkmalen und Label wird *Muster* genannt. Eine Menge aus Mustern wird Trainingsdaten(-satz) oder Trainingsmenge genannt. Mit Hilfe einer Trainingsmenge kann für ein Objekt ohne Label mit bekannten Merkmalen ein Label vorhergesagt werden.

Bei *nicht überwachtem Lernen* bleiben die Label der historischen Daten ungenutzt. Wenn vorhandenes Wissen ungenutzt bleibt, ist die Prognose potentiell schlechter und diese Methode damit un-

geeignet. Auch *bestärkendes Lernen* ist ungeeignet, weil dabei der Effekt der künstlichen Intelligenz auf die Außenwelt beobachtet wird, historische Daten aber nicht geändert werden können.

Übrig bleibt also das überwachte Lernen. Dessen in Frage kommende Algorithmen werden im Folgenden vorgestellt und hinsichtlich des Problems bewertet. Die Kriterien dafür sind:

- Geeignet zur Time-Series-Prediction
- Vorhersage kontinuierlicher Werte
- Akzeptable Laufzeit bei großen Datensätzen
- Umgang mit unvollständigen Datensätzen

Der Relief-Algorithmus sagt diskrete, keine kontinuierlichen Werte vorher und ist damit ungeeignet.

Bei den *Künstlichen Neuronalen Netzen* handelt es sich um Netze aus künstlichen Neuronen. Durch die Eingangsmuster und deren Ergebnisse wird ein künstliches Netz aufgespannt. Durch Abgleich der Ausgaben werden die Konfigurationen des Netzes angepasst, damit ein gutes Ergebnis zustande kommt. Des Weiteren können die KNN auch für Zeitreihenprognose verwendet werden, dies wurde beispielsweise in [Eis05] betrachtet.

Die *Regression* ist ein statistisches Analyseverfahren, in dem Beziehungen zwischen abhängigen und unabhängigen Variablen gezogen werden. Dabei handelt es sich um die lineare Regression und die Autoregression. Bei der linearen Regression wird die abhängige Variable über eine Linearkombination dargestellt. Durch diese Kombination soll das Modell aufgezogen werden. In der Autoregression dagegen sind die Datenpunkte geordnet, wie beispielsweise in einer Zeitreihe. Die vorhandenen Daten werden in diesem Fall als unabhängige Variablen verwendet und in Bezug zur abhängigen Variable gesetzt. Die Autoregression wurde im Weiteren jedoch nicht verwendet, da mehr Arbeiten zum Thema Time-Series-Prediction mit der linearen Regression recherchiert wurden und somit dafür der Aufwand geringer eingeschätzt wurde.

Bei der K-nearest neighbors (k-NN)-Regression werden die zu den vorherzusagenden Datensätzen ähnlichsten historischen Datensätze gesucht. Aus diesen wird eine Vorhersage generiert [Kel12], [PTK15].

Bei den *Support Vector Machines (SVM)* werden die Daten in Klassen unterteilt und die Klassen und Daten anschließend in einem Vektorraum dargestellt. Zwischen den Klassengrenzen werden jedoch Bereiche für die noch nicht zugewiesenen Daten bzw. Objekte im Vektorraum freigelassen. Die Support Vector Machines können auch zur Regression genutzt werden. Bei einer *Support Vector Regression* wird dieser Freiraum als Regression genutzt. Das Support Vector Machine-Verfahren kann deshalb sowohl zur Vorhersage diskreter, als auch numerischer Attribute eingesetzt werden.

6.7.1. Implementierung

Im Folgenden wird die Eigenimplementation einer k-NN-Regression vorgestellt. Sie war Teil des Prototyps und schneller umsetzbar, als das Recherchieren und Einbinden einer Bibliothek für maschinelles Lernen.

Zunächst werden die k Tage vom Wettermodul angefordert (Markt-PUC3), welche dem vorherzusagenden Tag (in Bezug auf das Wetter) am ähnlichsten sind (s. Abschnitt 6.6.2). k ist dabei ein einstellbarer Konfigurationsparameter.

Für die gelieferten Tage werden die Marktpreise aus den historischen Daten geladen. Diese Daten werden in Form von Instanzen der Klasse `MarketEntity` bereitgestellt. Der Durchschnitt pro Stunde der jeweiligen k Tage wird anschließend berechnet und stellt die Vorhersage der Marktpreise dar.

Die Vorhersage wird ebenfalls in Form einer `MarketEntity` bereitgestellt.

Die historischen Marktdaten befinden sich in der MySQL-Datenbank, werden jedoch direkt nach dem Starten der Anwendung geladen und als HashMap im Arbeitsspeicher gepuffert, damit nicht bei jedem Zugriff die relativ langen Zugriffszeiten der Datenbank den Vorgang unnötig verzögern. Bei einem Test hat das Puffern der ausgelesenen Marktdaten die Zugriffszeit von mehr als 5s auf 200ms verringert.

Nach Abschluss des Zwischenberichts hat sich die Projektgruppe nach ausführlicher Recherche zum Thema maschinelles Lernen für das Framework Weka (s. Abschnitt 5.13) entschieden und den selbst implementierten k-NN Algorithmus entfernt.

Für die Erstellung der Marktprognose (Markt-PUC6) eines Tages wird aus den n vorherigen Tagen gelernt, wobei n ein Konfigurationsparameter ist. Damit Weka aus den historischen Daten lernen kann, müssen diese in passender Form vorliegen. Dafür wird aus den Marktdaten und Wetterdaten der Datenbank n *WekaInstances* die Trainingsmenge, erzeugt. Von diesen lernt ein *WekaForecaster* ein Modell und erzeugt anschließend eine Prognose. Der *WekaForecaster* ist eine abstrakte Klasse, für die jeweils eine Implementierung, wie beispielsweise Lineare Regression oder Support Vector Regression, gewählt wird.

Die Vorhersage der Stundenpreise wurde auf zwei unterschiedliche Arten implementiert: Entweder werden die nächsten 24 Stundenpreise vorhergesagt, oder die Stundenpreise sind Attribute eines Tages und es wird ein Tag vorhergesagt.

Es wurden vier Vorhersage-Algorithmen getestet: IBk-Regression (k-NN-Regression), Linear Regression und die Support Vector Regression-Algorithmen SMOreg und LIBSVM. Linear Regression, IBk-Regression und SMOreg sind Weka-eigene Algorithmen. LIBSVM ist eine bekannte Bibliothek für Support Vector Machines der National Taiwan University.

Konfiguration Weka bietet die Funktion *overlay data* für Zeitreihen [Wek]. Mit dieser werden Daten über den Zeitraum der Trainingsmenge und der vorherzusagenden Tage gelegt, also die Merkmale der Muster um ein Merkmal erweitert. Diese Funktion verbessert die Vorhersagequalität und eignet sich ideal für Anwendungsfälle, bei denen die überlagerten Daten gut mit einem physikalischen Modell vorhersagbar sind.

Laut [GKLL+03] ist die Temperatur ein wesentlicher Einflussparameter für eine hohe Last im Winter und geringere Last im Sommer. Außerdem seien typische Tages- und Wochenzyklen sichtbar. Aus diesem Grund wurden die historischen Daten der Temperatur und Sonneneinstrahlung über die Trainingsmenge und die Wetterprognose von Temperatur und Sonneneinstrahlung über den vorherzusagenden Tag gelegt. Anstatt z. B. die durchschnittliche Tagestemperatur als Überlagerungs-Attribut

zu wählen, wurde pro Block ein Überlagerungs-Attribut mit der Temperatur dieses Blocks hinzugefügt.

Weiterhin kann Weka periodische Attribute vom Datum ableiten und als Überlagerungsdaten nutzen, wie beispielsweise Monatstag, Wochentag, Werktag/Wochenende, Quartal oder Monat des Jahres.

Kontextdaten wie Aktienkurse und Ölpreise sind oft relevant für den Strompreis [AM12, Seite 48]. Die Projektgruppe hat exemplarisch den Gaspreis von der Seite der European Energy Exchange mithilfe eine Web-Scrapers geladen und den Trainingsdaten hinzugefügt.

Da LIBSVM die *overlay data* von Weka nicht beachtet, wurde SMOreg als zweiter Support Vector Regression Algorithmus gewählt.

Es existiert kaum Dokumentation zur Benutzung der Weka Zeitreihenanalyse mittels Java-Code. Für die Funktion *overlay data* ist die fehlende Dokumentation besonders kritisch. Dementsprechend wurde versucht, den Code der Weka-GUI nachzuvollziehen und dessen Benutzung von *overlay data* nachzuahmen. Allerdings verfolgt der Weka-Code den Model-View-Prinzipien nur unzureichend, stattdessen ist sehr viel Logik im Code der grafischen Oberfläche, womit der Code schlecht wiederverwendbar ist. Zudem wirkt der Code unstrukturiert und schlecht nachvollziehbar, da er größtenteils unkommentiert ist. So ist beispielsweise die Methode zum Starten der Prognose (`weka.classifiers.timeseries.gui.ForecastingThread.ForecastingPanel.run()`) über 600 Zeilen lang.

Diese Schwierigkeiten sind so gravierend, dass stattdessen ein einfacherer Weg gewählt wurde: Die Einstellungen für die Prognose werden per Code in der Weka-GUI gesetzt und das Ergebnis der Prognose wird per Code aus der GUI ausgelesen. Dazu wurde der Quellcode von Weka z. B. um Getter für den Start-Prognose-Button erweitert. Abbildung Abbildung 26 zeigt die Benutzung der grafischen Oberfläche von Weka anhand der Prognose des Pakets *earlyMorning*. Damit während der Ausführung von *Powder* keine Weka-GUI erscheint, wird Weka *headless*, mit unsichtbarer grafischer Oberfläche ausgeführt.

Parameter der Support Vector Regression Für beide Support Vector Regression-Algorithmen wird empfohlen, den *RBF-Kernel* zu benutzen, da dieser für die meisten Probleme gut geeignet ist [Lin10]. Der Kernel ist ein wesentlicher Bestandteil der Support Vector Regression und dient der Mustererkennung. Nach [Lin10] wird empfohlen, für den RBF-Kernel die Parameter Gamma γ und *Cost* C anzupassen. Die *Cost* steuert die Kosten der falschen Klassifikation. Ein kleines C bedeutet eine kleine Bestrafung bei Falschklassifikation. Durch ein zu klein gewähltes C wird eine Unteranpassung (*engl.* underfitting) riskiert. Analog bedeutet ein großes C eine hohe Bestrafung, wodurch eine Überanpassung (*engl.* overfitting) begünstigt wird. Das Gamma bestimmt, wie weit der Einfluss eine einzelnen Trainingsmusters reicht. Ein kleines γ bedeutet einen weiten Einflussbereich, ein großes γ bedeutet einen engen Einflussbereich. Demnach ist γ das Inverse des Einflussradius eines Individuums.

Die Parameter C und γ müssen daher für jede Datenmenge individuell angepasst werden. Diese können experimentell mittels K-nearest neighbors ermittelt werden. Zunächst muss festgelegt werden, welche Werte für C und γ relevant sein könnten. Nach [BLB+16] wird hierfür empfohlen, eine glsglos:grid-search anhand einer logarithmischen Skala durchzuführen. Durch experimentelle Be-

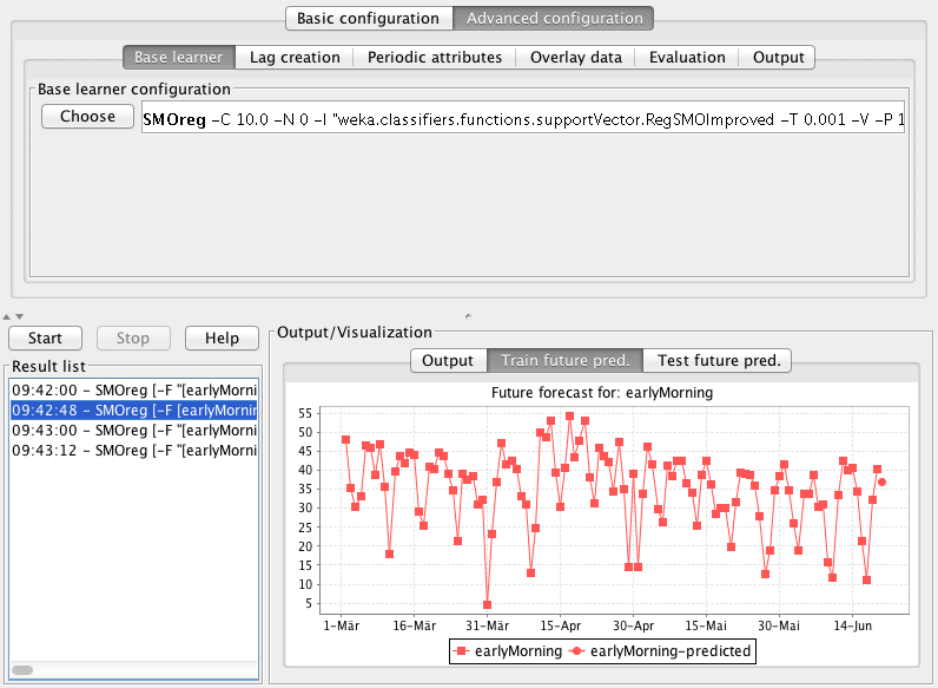


Abbildung 26: Historische Daten und Vorhersage mit Weka

stimmung auf der Datenmenge mit LIBSVM können die Intervallgrenzen auf $10^{-12<x<12}$ festgelegt werden. Zur Reduzierung der Laufzeit wird außerdem anstatt einer *grid-search* ein Latin Hypercube verwendet. Eine grid-search kombiniert alle C-Werte mit allen γ-Werten. Dadurch beträgt die *grid-search*-Laufzeit $O(n^2)$. Bei Latin Hypercube werden Parameterkombinationen so festgelegt, dass der Parameterraum gut abgedeckt wird. Die Laufzeit wird auf $O(n)$ reduziert. Eine detailliertere Beschreibung des Latin Hypercube befindet sich unter 6.10.1

Bei der Kreuzvalidierung wird zwischen einer einfachen und n-fachen Kreuzvalidierung unterschieden. Bei der einfachen Kreuzvalidierung wird die Trainingsmenge in Trainingsmenge und *Validierungsmenge* aufgeteilt. Bei der Validierungsmenge werden zunächst die vorhandenen Labels ignoriert. Sie werden mittels der Trainingsmenge versucht, vorherzusagen. Mit Hilfe der bekannten Label wird dann die Abweichung von Vorhersage zu realem Label berechnet. Im Unterschied zur einfachen Kreuzvalidierung wird bei der n-fachen Kreuzvalidierung der Datensatz in n Teile unterteilt. $\frac{n-1}{n}$ Teildaten werden als Trainingsmenge benutzt, $\frac{1}{n}$ wird als Validierungsmenge verwendet. Die Kreuzvalidierung wird so häufig ausgeführt, dass jeder Teildatensatz Validierungsmenge war. Die n-fache Kreuzvalidierung vermeidet häufig besser eine Überanpassung, als die einfache. Für eine n-fache Kreuzvalidierung auf Zeitreihen muss der Datensatz in Jahre aufgeteilt werden. Dies ist für *Powder* nicht möglich, da auch mit den Wetterdaten gelernt werden soll und diese nur für 2012 verfügbar sind. Daher kann nur eine einfache Kreuzvalidierung durchgeführt werden.

Für die Messung der Güte eines Algorithmus' ist der Root Mean Square Error (RMSE) eine gute Wahl. Er ist die Quadratwurzel des durchschnittlichen Prognosefehlers und wird berechnet, indem für alle prognostizierten Blockpreise eines Tages die Abweichung zu den tatsächlichen, historischen Blockpreisen berechnet wird. Die Wurzel des Durchschnitts dieser Abweichungen ist der RMSE.

Um Algorithmen miteinander zu vergleichen, wird der durchschnittliche RMSE über einen großen Zeitraum gebildet.

Für jedes Parameterpaar (auch Parameterset genannt) aus Gamma und Cost wird eine Kreuzvalidierung durchgeführt und für die gesamte Validierungsmenge der durchschnittliche RMSE gebildet. Das Parameterpaar, das zum niedrigsten RMSE führte, wird gespeichert.

6.7.2. Auswahl eines Algorithmus

In diesem Abschnitt werden die Messergebnisse der Güte der verschiedenen Algorithmen und der Konfigurationsparameter vorgestellt.

In Abbildung Abbildung 27 wurde die Auswirkung verschiedener *overlay data* berechnet. Dafür wurden Marktprognosen mit dem Algorithmus SMOreg (Support Vector Regression) für alle Tage im Zeitraum vom 20.06.2012 bis zum 20.12.2012 erstellt und jeweils der RMSE berechnet. Gelernt wurde für jeden Tag anhand der letzten 110 Tage. Der Durchschnitt dieser RMSE-Werte gibt die Güte an. beispielsweise ist der erste Wert in der Abbildung die Differenz aus der Güte mit Temperatur-Overlay und ohne Temperatur-Overlay. In *Powder* werden alle Overlays genutzt, welche den RMSE verringern. Lediglich die Sonneneinstrahlung erhöht den RMSE. Vermutlich, weil die Tag- und Nachtwerte so stark schwanken, dass die Time-Series-Algorithmen diese schlecht verarbeiten können.

	Auswirkung auf Ø RMSE bei	
Overlay data	Blöcken in €	Stunden in €
Temperatur	+0,007	+0,03
Sonneneinstrahlung	+12,26	+14,91
Periodische Attribute	+0	+0
Gaspreis	+0,14	+0,09

Abbildung 27: Auswirkung auf die Abweichung verschiedener *overlay data* über den Zeitraum 20.06.2012 bis 20.12.2012

Wie in Abbildung Abbildung 28 zu sehen ist, hat bei den k-NN Algorithmen der Wert des Parameters k (für die Anzahl der nächsten Nachbarn) nur geringe Auswirkungen auf das Ergebnis.

Algorithmus	Ø RMSE in €
Selbst implementierter k-NN mit k=3	17,16
Selbst implementierter k-NN mit k=10	16,33
Selbst implementierter k-NN mit k=15	16,53
IBk-Regression (k-NN-Regression) mit k=3	6,06
IBk-Regression (k-NN-Regression) mit k=10	6,64
IBk-Regression (k-NN-Regression) mit k=15	7,03

Abbildung 28: Abweichung über den Zeitraum 20.06.2012 bis 20.12.2012 bei einer Trainingsmenge von 110

In Abbildung Abbildung 29 werden alle ausgewählten Algorithmen miteinander verglichen. Dafür wurden die zuvor berechneten optimalen Konfigurationsparameter (k, Gamma, Cost) benutzt. Auffällig ist, dass MultilayerPerceptron eine sehr hohe Laufzeit hat und miserable Ergebnisse liefert.

Der k-NN Algorithmus hingegen läuft sehr schnell. Das beste Ergebnis liefert SMOreg, ein Support Vector Regression Algorithmus.

Algorithmus	Ø RMSE Blöcke in €	Ø RMSE Stunden in €
Selbst implementierter k-NN	17,35	16,98
IBk-Regression (k-NN-Regression)	6,14	5,98
Linear Regression	8,43	7,58
SMOreg (Support Vector Regression)	5,27	4,81
LIBSVM (Support Vector Regression)	9,28	9,46
MultilayerPerceptron (Neuronales Netz)	$9,00 \cdot 10^{80}$	$3,35 \cdot 10^{109}$

Abbildung 29: Abweichung über den Zeitraum 20.06.2012 bis 20.12.2012 bei einer Trainingsmenge von 110, bzw. über den Zeitraum 20.06.2012 bis 20.07.2012 und einer Trainingsmenge von 30 bei MultilayerPerceptron

Eine Erhöhung der Trainingsmenge hat, wie erwartet, eine Verbesserung der Güte zur Folge (s. Abb. 30).

Trainingsmenge	Ø RMSE in €
30	5,99
110	5,04
330	5,02

Abbildung 30: Abweichung über den Zeitraum 20.06.2012 bis 20.12.2012 mit SMOreg

In der Masterarbeit „A Novel Algorithmic Trading Framework Applying Evolution and Machine Learning for Portfolio Optimization"[AM12] werden verschiedene Algorithmen für die Prognose des Kapitalmarktes verwendet und miteinander verglichen. Daraus geht hervor, dass sich besonders Support Vector Machines und Support Vector Regression anbieten. Diese Erkenntnis deckt sich mit dem Ergebnis der Projektgruppe.

6.7.3. Umsetzung der Anforderungen

Die Anforderung *Marktdaten aktualisieren* (Markt-PUC1) wurde verworfen, weil die historischen Wetterdaten auf das Jahr 2012 beschränkt sind und damit auch nur die Marktdaten für 2012 notwendig sind.

Die Anforderungen *Marktdatenfilter erstellen* (Markt-PUC4) und *Marktdatenfilter anwenden* (Markt-PUC5) sollten die Marktdaten auf Datenbankebene nach relevanten Tagen für den k-NN Algorithmus filtern. Diese Anforderungen wurden verworfen, weil der Zugriff auf die Marktdaten durch Zwischenspeicherung im Arbeitsspeicher sehr schnell ist. Zudem ist das Abfragen von *MarketEntity*-Datenobjekten per Datum ausreichend.

Für die Marktprognose werden historische Marktdaten verwendet. Das Optimierungsmodul verwendet die Marktprognose in der Optimierung. Damit ist die spezifische Anforderung *historische Daten für die Optimierung verwenden* (L-15) erfüllt.

6.8. Flexibilitätenmodul

Dieses Kapitel beschäftigt sich mit der Erzeugung von Flexibilitäten und setzt die Anforderung Flex-PUC4 um. Dazu werden Informationen von physischen Anlagen benötigt, um Anlagenmodelle zur Erzeugung von Fahrplänen zu generieren, die anschließend aggregiert werden und als Flexibilitäten dargestellt werden. Dazu wird in dem Flexibilitätenmodul für jede Anlage eine Instanz erstellt, wodurch die Anforderung Flex-PUC3 erfüllt ist. Die Benötigten Informationen bestehen aus deren Stammdaten, Status und Einschränkungen. Da in diesem Kapitel einige themenspezifische Begriffe genannt werden, wird zunächst ein Grundlagen- und Entscheidungsabschnitt folgen. Anschließend wird auf die Simulation und Simulationsumgebung eingegangen und mit Entwurf und Implementierung der Anlagentypen fortgefahren. Das Kapitel endet mit einer ausführlichen Kosten- und Erlösberechnung der Fahrpläne. Für die Berechnung der Flexibilitäten werden jedoch nur Anlagenmodelle für die Energieerzeugung berücksichtigt, allerdings werden keine Verbraucher elektrischer Energie einbezogen. Dies kann damit begründet werden, dass dadurch die im vorherigen getroffene Anforderung L-06 erfüllt werden kann.

6.8.1. Grundlagen und Entscheidungen

Flexibilität Eine Flexibilität stellt eine Ansammlung von realisierbaren Fahrplänen einer Anlagen dar. Einige Anlagentypen können im Tagesverlauf unterschiedliche Leistungsverteilungen haben, sodass es möglich ist, die erzeugte elektrische Energie zu unterschiedlichen Zeitpunkten bereitzustellen. Jede dieser Leistungsverteilungen repräsentiert einen Fahrplan der Anlage.

Anlagenarten Im Rahmen des Projekts wurde die Anforderung gestellt, *Photovoltaikanlagen* und *Blockheizkraftwerke* als Anlagentypen zu verwenden. Aus Gründen der Komplexität wurde die *Batterie* als Anlagentyp nicht umgesetzt. Daher wurde auch die Anforderung L-03 zunächst nicht umgesetzt. Diese Aufgabe müsste daher im Anschluss an das Projekt erfolgen. Damit leichter verstanden werden kann, worum es sich dabei handelt, werden diese beiden Anlagentypen als nächstes kurz erläutert.

Bei *Photovoltaikanlagen*, im Folgenden PV-Anlagen genannt, handelt es sich um elektrische Energieerzeuger, welche die Sonne als Energiequelle nutzen. Die PV-Anlagen gehören zu den Energieerzeugern aus erneuerbaren Energien, da diese keine endlichen Ressourcen benötigen. Sie bestehen aus einer Reihe von Solarmodulen, die wiederum aus einer großen Anzahl von Solarzellen bestehen. Trifft ein Sonnenstrahl auf solch eine Solarzelle, spalten die darin enthaltenen Photonen die Elektronen aus ihrer Gitterstruktur. Dabei wird beispielsweise Silizum als Elektronenträger verwendet. Dabei entsteht ein positiv geladenes Loch und das frei bewegliche, abgespaltene, negative Elektron, welches frei beweglich in den PN-Übergang der Solarzelle wandert. Bei dem Loch handelt es sich nun um den freigewordenen Platz in der Gitterstruktur des Stoffes, welcher vorher das Elektron beinhaltet hat. Der PN-Übergang beschreibt des Weiteren einen Raum, der zwischen einer positiv und einer negativ dotierten Seite liegt. In diesem Übergang wird das Loch-Elektronen-Paar wieder gespaltet und an die Kontakte der Solarzelle abgegeben, welche den entstandenen Strom dann wiederum an das Solarmodul und dann an die Wechselrichter weiterleiten. Diese wandeln den Strom aus Gleich-

strom in Wechselstrom um, damit dieser ins Netz eingespeist werden kann. PV-Anlagen sind zumeist auf Hausdächern oder auf Freiflächen zu finden. Es gibt dabei unterschiedliche Modulvarianten, wie Dünnschicht- oder Dickschichtmodule, sowie die Unterscheidung in multikristallinen Zellen oder monokristallinen Zellen [Qua11].

Die *Blockheizkraftwerke* (BHKWs) stellen modular aufgebaute Anlagen dar, mit deren Hilfe elektrische und thermische Energie erzeugt werden kann. Dabei verwenden BHKWs das Prinzip der Kraft-Wärme-Kopplung, welche die Gewinnung von mechanischer und thermischer Energie ist, wobei die mechanische in elektrische Energie umgewandelt wird. Diese Anlagen müssen den Stromerzeuger antreiben, dazu können beispielsweise Verbrennungsmotoren, Gasturbinen oder Stirlingmotoren verwendet werden. Je nachdem, welcher Antrieb verwendet wird, müssen unterschiedliche Kraftstoffarten eingesetzt werden. Jede dieser Kraftstoffarten liefert eine gewisse Menge an Energie, die nach Gesamtwirkungsgrad der Anlage besser oder schlechter ausgenutzt wird. Der Anteil der erzeugten thermischen Leistung wird häufig in Pufferspeichern zwischengelagert, während die elektrische Leistung in das Netz eingespeist oder für den Eigenverbrauch verwendet wird. Durch den thermischen Verbrauch im Haushalt oder einem Fernwärmenetz entleert sich der Pufferspeicher und neue thermische und elektrische Leistung kann durch das BHKW erzeugt werden. Im Rahmen der Projektgruppe wird der Haushaltsverbrauch simuliert, um ein Lastprofil zu erhalten und die Erzeugung der thermischen Leistung zu simulieren. Die Berechnung dafür wurde bereits im Abschnitt 6.4 betrachtet.

Auswahl der Anlagenmodelle Wie bereits in der Einleitung beschrieben, ist es der Projektgruppe erlaubt simulierte Anlagen zu verwenden. Dieses wird in AnforderungL-19 beschrieben, welche durch die Implementierung der Anlagenmodelle erfüllt wird. Dazu ist es notwendig, dass es für jeden Anlagentyp ein Modell gibt, welches dessen Funktionsweise beschreibt. Für die Modellierung von Energieerzeugern kann auf eine Reihe von bereits vorhandener Software und Modelle zurückgegriffen werden. Damit eine sinnvolle Lösungsmöglichkeit für die Projektgruppe ausgewählt werden konnte, wurde eine Evaluation über eine Auswahl an Optionen durchgeführt. Ein besonderes Augenmerk wurde dabei auf eine mögliche Modellierung und anschließende Simulation der Anlagen in der Software TwinCAT der Firma Beckhoff gelegt. Dabei handelt es sich um einen, an der Universität Oldenburg bereits vorhandenen, Programmable Logical Controller (PLC), auf denen die Software TwinCAT läuft. Dadurch ist es möglich, die PLCs als physische Anlagen anzusehen, auf der das Anlagenmodell implementiert wurde und mit der über Kommunikationsprotokolle kommuniziert werden kann. Eine genauere Betrachtung des Beckhoff-Systems und der Möglichkeiten kann dem beigefügten Seminarband G entnommen werden. TwinCAT bietet dabei drei relevante Möglichkeiten, wie die Modelle auf den PLCs angesprochen und unter welchen Anforderung die Modelle implementiert werden können. Neben der Verwendung der TwinCAT-Software wurde für die Evaluation auch die Möglichkeit der eigenen Modellierung, ohne Verwendung der PLCs und TwinCAT, betrachtet. Die Evaluation erfolgte dabei über die Betrachtung der Vor- und Nachteile jeder Möglichkeit, die in der Tabelle 2 abgebildet werden.

Die Entscheidung wurde nach der Evaluation für die eigene Entwicklung gefällt. Die Hauptkriterien für die Entscheidung waren vor allem die frei wählbare Programmiersprache sowie, dass es keine

Möglichkeit	**Vorteile**	**Nachteile**	**Schnittstellen**
TwinCAT mit C# oder .Net	- Verwendung TwinCAT Debugging u. Umgebung - TwinCAT Runetime - Bausteine von TwinCAT	- Keine Echtzeitfähige Programmierung - State Machine muss beachtet werden - Vorgegebene Schnittstellen - Schlechte Dokumentation	ADS-Schnittstelle o. OPC UA verwendbar
TwinCAT C/ C++ oder IEC 61131-3	- Leistungsstarke Programmiersprache - Einsatz bei Programmierung für Maschinen - Echtzeitfähig - Beispiele vorhanden	- Nicht alle möglichen Features von C++ möglich - Keine Verwendung von externen Bibliotheken - IEC 61131 strikte Programmierung - State Machine	ADS-Schnittstelle o. OPC UA verwendbar
TwinCAT Matlab/ Simulink	- Zusätze für elektr. Systeme - Entw. komplexer Systeme - Grafische Programmierung - Vielfalt an Toolboxes	- Einarbeitung in Programmiersprache - Nur TwinCAT Bibliotheken nutzbar - State Machine	ADS-Schnittstelle o. OPC UA verwendbar
Eigene Entwicklung	- Selbst auswählbare Programmiersprache - PLCs müssen nicht verwendet werden - Keine Einschränkung durch TwinCAT - Viele Beispiele für unterschiedliche Sprachen vorhanden	- Echtzeitsimulation schwierig Umsetzbar - Tools von TwinCAT nicht verwendbar - Höherer Aufwand	Schnittstellen frei wählbar

Tabelle 2: Vor- und Nachteile der Anlagenmodelle

Einschränkungen durch ein anderes Tool gibt. Außerdem können die eigenen Modelle gut für die Anforderungen von verschiedensten Simulationsumgebungen angepasst werden. Auch die Vielzahl an bereits vorhandenen Beispielen für Anlagenmodelle für verschiedene Programmiersprachen, die für die Anforderungen der Projektgruppe angepasst werden können, waren entscheidende Faktoren für die Entscheidung. Zusätzlich entfällt der Punkt, dass Echtzeitsimulationen schwierig umsetzbar sind, da dies keine Anforderung an das System ist. Ebenso ist der erhöhte Aufwand nur auf die eigene Umsetzung bezogen und gleicht sich aus, wenn man betrachtet, dass die Auswahl einer bekannten Programmiersprache die Einarbeitungszeit erheblich verkürzt.

Simulationsumgebung Neben der Auswahl der geeigneten Software zur Erstellung der Anlagenmodelle ist es ebenfalls notwendig eine Simulationsumgebung zu nutzen, in der das Modell ausgeführt und simuliert werden kann. Für die Anforderungen der Projektgruppe ist es dabei wichtig, dass bei der Simulation in der Umgebung auch Flexibilitäten der zu simulierenden Anlagen erstellt werden können. Wie bei der Software für Anlagenmodelle gibt es auch bei den Simulationsumgebungen eine Vielzahl von Softwaresystemen. Das Vorgehen für die Auswahl der Simulationsumgebung wurde äquivalent zur Auswahl der Software für die Anlagenmodellierung durchgeführt. Nach einer durchgeführten Recherche über mögliche Simulationsumgebungen wurden die Umgebungen Power Factory, Mosaik, Ptolemy II, Mirabel, OpenMUC, EnergyPlus sowie eine Eigenentwicklung in die Evaluation aufgenommen. Die Darstellung der Evaluation erfolgt in der Tabelle 3 mit den Vor- und Nachteilen der Simulationsumgebungen. Zum besseren Verständnis der Auswahl einer dieser Umgebungen wird zunächst jede vorgestellt.

Power Factory ist das kostenpflichtige Simulationstool der Firma DIgSILENT GmbH. Das Haupteinsatzgebiet dieser Software ist das Planen, Entwickeln und Analysieren von Versorgungsnetzen. Dabei setzt DIgSILENT in seiner Software auf eine grafische Oberfläche, in der über visuelles Programmieren Netzkomponenten generiert und Verbindungen untereinander hinzugefügt werden können. Beim Hinzufügen der Komponenten können auch eigenentwickelte Anlagenmodelle verwendet werden. Die Modellierung der Anlagenmodelle erfolgt äquivalent zu der Programmierung mit Matlab Simulink, weshalb auch Simulink verwendet werden kann. Die erstellten Netze können mit Hilfe der Software durch Lastfluss- sowie Kurzschlussberechnungen und Transientenanalyse überprüft, sowie zusätzlich durch Netzoptimierungen optimiert werden. Vorteile bietet die Simulationsumgebung Power Factory durch die grafische Programmierung, sowie die Integration von Anlagenmodellen. Dagegen ist die Software eher auf Netzberechnungen ausgelegt, was nicht Aufgabe der Projektgruppe ist. Außerdem ist unklar, ob Flexibilitäten berechnet werden können und auch die Kosten der akademischen Lizenz von 150 Euro ist ein weiterer negativer Punkt.

Die Simulationsumgebung *Mosaik* wurde von dem der Universität Oldenburg nahen Forschungsinstitut OFFIS entwickelt. Das Tool zielt darauf ab, vorhandene oder eigens erstellte Anlagen- und Netzkomponentenmodelle über eine Co-Simulation zu verbinden und in einer Umgebung zu simulieren. Dabei können die Modelle in viele gängige Programmiersprache, wie *Python*, *Java* oder *C++* geschrieben und über eine API leicht an Mosaik angebunden werden. Zudem können diese Modelle unterschiedliche zeitliche Schritte (diskrete Simulation) verwenden, da Mosaik diese über eine Kommunikation mit jedem Modell diese Schritte einzeln vorgehen lassen kann. Die Mosaik-Umgebung

wurde in der Programmiersprache Python entwickelt, besitzt aber beispielsweise für Modelle in Java eine eigene API. Als Vorteile bietet Mosaik im Rahmen der Projektgruppe, dass die Modelle der Anlagen in Java entwickelt werden können. Weiterhin wurden bereits Programmierbausteine für Suchraummodelle und Flexibilitäten von Anlagen für den Einsatz in Mosaik entwickelt. Außerdem bietet Mosaik eine einfach Integration von Evaluationsmethoden (Latin Hypercube) und eine LPGL (GNU Lesser General Public License) Lizenz, bei welcher es sich um eine freie Lizenz für Software handelt. Als negative Punkte können die Unübersichtlichkeit durch gleich benannte Klassen sowie die Integration des Java-Scripts in das Python-Modul genannt werden.

Die Simulationsumgebung *Ptolemy II* wurde von der Berkeley Universität entwickelt, um die steigende Anzahl von heterogenen Systemen zu modellieren. Dabei setzt Ptolemy auf Cyber-physikalische Systeme. Die Software besteht aus einem Kern von Java Klassen und Paketen, sowie einer Vielzahl von Integrationsmöglichkeiten von anderen Sprachen, wie Matlab und Python. Ptolemy ist ein graphisches Programmiertool mit einer eigenen graphischen Oberfläche. Als Vorteile bietet Ptolemy eine Vielzahl an Projektdemos, sowie die Arbeit mit unterschiedlichen Programmiersprachen. Des Weiteren können sowohl diskrete, als auch kontinuierliche Simulationen verwendet werden und das Simulationstool ist als Open Source Software verfügbar. Negativ ist, dass auch für dieses Tool nicht bekannt ist, ob Flexibilitäten berechnet werden können. Außerdem braucht das Tool eine große Einarbeitungszeit. Zudem erscheint die Integration anderer Programmiersprachen sehr komplex. Das Auslesen von Daten ist meistens auch nur über Graphiken möglich.

Das *Mirabel-Projekt* hat sich der Problemstellungen der Micro-Request-Based Aggregation, Forecasting and Scheduling of Energy Demand, Supply and Distribution angenommen. Das Tool bietet zudem eine mögliche Variante für die Erstellung von Flexibilitäten. Da jedoch die Einbindung von unterschiedlichen Anlagenmodellen, sowie die Struktur der Flexibilitäten ungenau beschrieben wurden und dies eine erhebliche Einarbeitungszeit in das Tool bedeutet, wurde die Simulationsumgebung nicht weiter betrachtet.

Als weiteres Framework wurde *OpenMUC* des Fraunhofer ISE betrachtet, welches Projekte von virtuellen Kraftwerken in Einfamilienhäusern unterstützt. Das Framework arbeitet dabei mit Java, OSGi und zielt auf das Überwachen und Monitoring der Komponenten ab. Dabei wird auf eine Anzahl von bereits integrierten Kommunikationsprotokollen, wie OPC UA usw., zurückgegriffen. Nach der Vorstellung des Tools wurde in der Projektgruppe entschieden, dass lediglich diese Protokolle von Bedeutung sind, weshalb diese in dem Bereich der internen und externen Kommunikationsschnittstellen noch einmal einer genaueren Betrachtung unterzogen werden (s. Unterabschnitt 6.2).

Das letzte betrachtete und bereits vorhandene Tool wurde vom U.S. Department of Energy mit dem Namen *EnergyPlus* entwickelt. Mit der Software werden vor allem Energieverbräuche im Bereich von Wärme und Elektrizität berechnet und simuliert. Das Tool wird daher von Architekten, Ingenieuren und Forschern zur Simulation von Temperatur, Dämmung, Wasserverbrauch, Stromverbrauch etc. in Gebäuden verwendet. Da in der Projektgruppe die Modellierung von Verbräuchen in Haushalten usw. nur ein kleiner Aufgabenbereich ist und das Tool keine Flexibilitäten berechnen oder mit diesen arbeiten kann, ist dieses Tool für die Zwecke der Projektgruppe nicht verwendbar. Daher wurde dieses auch nicht weiter betrachtet.

Neben den bereits entwickelten Simulationsumgebungen wurde auch die Möglichkeit der Eigenentwicklung betrachtet. Dabei fällt vor allem die Anpassung auf eigene Anforderungen, sowie die Auswahl der Programmiersprache und die Anpassung auf neue Anforderungen positiv ins Gewicht. Jedoch bedeutet die Eigenentwicklung auch einen hohen Aufwand. Dennoch sind viele der vorher genannten Simulationsumgebungen sehr komplex, sodass diese viel Zeit für die Einarbeitung benötigen.

Die Projektgruppe entschied sich bei einer Vorauswahl auf Basis der Tabellenwerte für die Eigenentwicklung und die Simulationsumgebung Mosaik. Dabei wurde Mosaik vor allem wegen der bereits vorhandenen Vorkenntnisse einer Kleingruppe in der Projektgruppe bezüglich der Modellierung der Anlagen mit Java und der Integration von Samples als Flexibilitäten gewählt. Die Eigenentwicklung wurde wegen der frei wählbaren Programmiersprache sowie der guten Anpassung an die Anforderungen der Projektgruppe gewählt. In einer zweiten Evaluation wurden die Möglichkeiten beider Varianten erneut evaluiert, dabei wurden besonders die schon vorhandenen Klassen für die Integration von Flexibilitäten und Anlagenmodellen, die in Kombination mit Mosaik verwendet werden können, als sehr positives Kriterium gesehen. Da Mosaik jedoch eine Vielzahl von Funktionen besitzt, die für die Anforderungen der Projektgruppe nicht notwendig sind, wurde sich für die Entwicklung einer eigenen Simulationsumgebung entschieden. Dabei hat sich die Projektgruppe jedoch eng an den Funktionen und den Strukturen von Mosaik orientiert, sodass die Möglichkeit der Verwendung einiger Klassen in Mosaik offen bleibt. Diese Möglichkeit bietet den Vorteil, die Umgebung nach den Bedürfnissen der Projektgruppe zu gestalten, aber auch bereits vorhandene Funktionen verwenden zu können.

6.8.2. Allgemeine Struktur

Dieser Abschnitt soll einen einleitenden Überblick über die Struktur des Flexibilitätenmodul liefern. Dazu wird zunächst auf die Umsetzung der eigenen Simulationsumgebung eingegangen und im Anschluss die Klassenstruktur visuell dargestellt. Da im Rahmen der Projektgruppe eine Day-Ahead Planung erfolgt, wurde die Simulationsumgebung darauf ausgelegt ohne überflüssige Aspekte, wie z. B. die reaktive Einsatzplanung. Dadurch wird auch die Anforderung L-18 betrachtet, jedoch nicht erfüllt.

Im `FlexibilityService` stehen RMI-Kommunikationsschnittstellen zu der Simulationsumgebung bereit. Diese bieten die Möglichkeit die Simulationsumgebung mit einer Auswahl an Anlagenmodellen zu initialisieren. Diese Auswahl wird nach Anlagentypen und Anlagenarten sortiert und getrennt voneinander einem `Sampler` übergeben. Diese Sampler Klassen greifen auf die Anlagenmodelle zu und simulieren für jeden äquidistanten Zeitpunkt eines Tages, wie beispielsweise alle 15 Minuten, den Zustand der Anlage. Nach erfolgreicher Simulation wird als Ergebnis ein Fahrplan ermittelt, der einen möglichen Leistungsverlauf der Anlage repräsentiert. Es gibt drei verschiedene Sampler-Typen, auf die im Kapitel 6.8.6 näher eingegangen wird. Durch diese Struktur wird gewährleistet, dass alle Anlagenmodelle in der Simulationsumgebung simuliert werden können. Ebenso lassen sich beliebig viele und unterschiedliche Anlagenmodelle hinzufügen, da jedes neue Modell nur die Klasse `Aplant` erweitern muss. Das in Abbildung 31 dargestellte Klassendiagramm liefert einen Überblick der Modelle, die im Rahmen der Projektgruppe implementiert wurden. Im Nachfolgenden soll die Abbildung anhand der Klasse `CHP_LPG` genauer betrachtet werden. Diese Klasse stellt

Name	Program-miersprache	Vorteile	Nachteile	Schnittstellen zu den Anlagen	Vorkennt-nisse
Power Factory	Matlab und weitere	- Simulation von kompletten Netzen - Anlagen können direkt entwickelt werden - Vielzahl von Netzberechnungsmöglichkeiten	- Kostenpflichtig - Auf Netzberechnungen ausgelegt - Sehr komplexes Tool - Viel Einarbeitungszeit	Können direkt im Tool entwickelt oder aus Matlab importiert werden	keine
Mosaik	Python, Anlagenmodelle können in einer Vielzahl von Sprachen geschrieben werden	- Vorgefertigte Funktionen - Evaluation (z.B. Latin Hypercube) - Sind eingearbeitet - Maven Projekt - LGPL Lizenz - Wiederverwendbar, Erweiterbar	- Unübersichtlich wegen gleich benannter Klassen - Integration von Java in Python	APIs vorhanden	Teil-nehmer Energie-praktikum
Ptolemy	Java (Matlab, Python möglich)	- Verwendet Java - Hierarchische Struktur von Entitäten - Grafischer Editor, wie Simulink - Integration von Matlab und Python - Modellierung von diskreten oder kontinuierlichen Systemen - Modellierung elektrische Systeme - Open Source	- Erfordert viel Einarbeitungszeit - Bedingungen an Integration anderer Skripte - Lückenhafte Dokumentation - Auslesen von Daten teilweise nur über Grafiken möglich - Keine Kenntnisse über Einbindung von Flexibilitäten	Modelle müssen in Ptolemy erstellt werden	Torben (Hiwi-Stelle)
Energy Plus	Keine Angabe	- Tool zur Berechnung von Verbräuchen	- Kenntnisse über Flexibilitäten nicht vorhanden - Einbindung von Anlagen nicht erkennbar - Haupteigenschaften des Tools nicht relevant für PG	Kein Wissen über Integration von Modelle	Keine
Mirabel	Keine Angabe	- Möglichkeit der Flexibilitätsberechnung	- Die Anbindung von eigenen Anlagenmodellen erscheint als schwierig	Kein Wissen über Integration von Modellen	Keine
Open MUC	Java, OSGi	- Kommunikationsschnittstellen - Anbindung für PV-Anlagen vorhanden - VK bereits vorhanden	- Für das Monitoring und Optimierungen in Haushalten verwendet	Anlagen über Schnittstellen gut integrierbar	Keine
Eigene Simula-tions-Umge-bung	Selbst wählbar	- Gut auf Anforderungen anpassbar - Einbindung von neuen Anforderungen leichter	- Hoher Aufwand - Könnte zweites Projekt werden - Scheduler selbst zu entwickeln	Kann von der PG bestimmt werden	Program-mierspra-che

Tabelle 3: Vergleich von Simulationsumgebungen

ein BHKW dar, welches durch den Kraftstoff LPG (Flüssiggas) betrieben wird. Sie wird von der Klasse `AGlobalCHP` abgeleitet, in der die Möglichkeit bereitgestellt wird allgemeingültige Methoden für BHKWs zu implementieren. Noch abstrakter ist die Ebene `Aplant`, in dieser Klasse können Methoden für alle Anlagenmodelle implementiert werden. Äquivalent gilt dies für die Klassen `PV` , `CHP_LPG_Techdata`, `PVTechData`, `CHPStatus` und `PVStatus`.

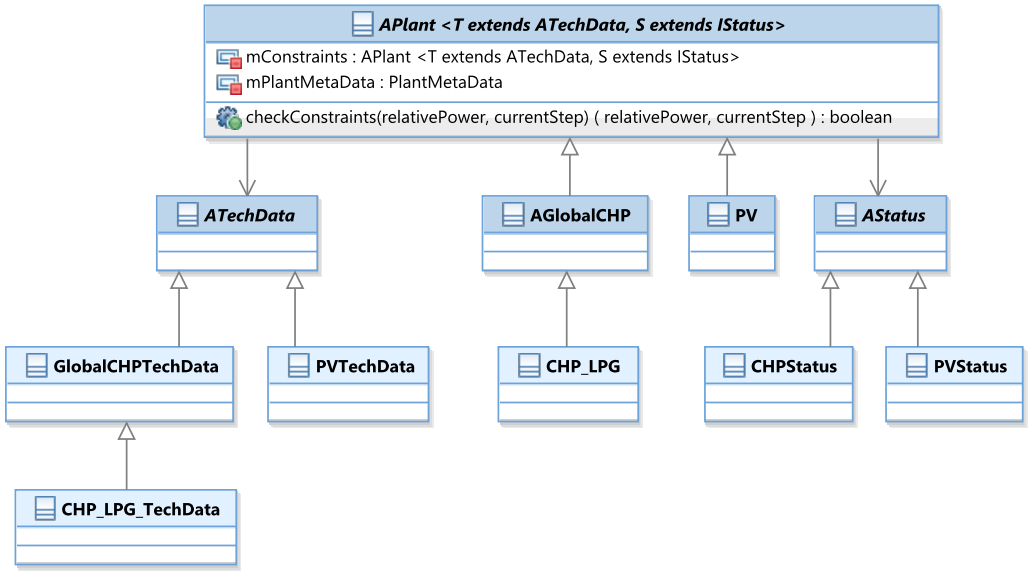


Abbildung 31: Modellstruktur der Anlagen

6.8.3. Entwurf Simulationsmodell PV-Anlage

Nachdem die Wahl auf die Erstellung eigener Anlagenmodelle mit einer frei wählbaren Programmiersprache gefallen ist (siehe vorheriges Kapitel), hat sich die Projektgruppe mit der Umsetzung der Modellierung der Anlagentypen befasst. Wie bereits häufiger erwähnt, sollen zunächst BHKWs und PV-Anlagen modelliert werden. In diesem Abschnitt wird die theoretische Berechnung der Photovoltaikanlagen betrachtet, welche im späteren Verlauf der Projektgruppe im Projekt implementiert wurde. Wie bei jedem Modell ist es auch beim PV möglich, dieses sehr komplex auf Basis der Neutronen und Elektronenbewegungen sowie der Entstehung der PN-Junktion zu modellieren. Da jedoch das Hauptaugenmerk auf der Optimierung des Portfolios liegt, wurde bei der Modellierung der PV-Anlagen ein einfaches Modell umgesetzt. Die gesamte Modellierung basiert auf Formel 1:

$$P[W] = G[W/m^2] \cdot \eta \cdot A[m^2]$$

Formel 1: Leistungsberechnung bei einer PV-Anlage

wobei

- P Leistung zu einem ausgewählten Zeitpunkt
- G Sonneneinstrahlung auf das Modul
- η Wirkungsgrad des Moduls
- A Fläche des Moduls

Die Formel wurde aus dem Buch „Regenerative Energiesysteme“ entnommen (siehe [Qua11]]. Für die Modellierung sind also drei Parameter von Bedeutung. Bei dem Wirkungsgrad und der Fläche der Anlage handelt es sich um Werte, die spezifisch für jede Anlage gewählt werden, sich aber während der Berechnung nicht mehr ändern. Diese werden zu den Stammdaten der einzelnen PV-Anlagen gezählt. Der Wirkungsgrad unterscheidet sich dabei in dem ausgewählten Material. Dabei zeigen monokristalline Zellen den höchsten Wirkungsgrad vor polykristallinen- und Dünnschichtzellen. Die Fläche wird über die Anzahl der verwendeten Module berechnet. Module sind dabei ein Zusammenschluss von mehreren Zellen. Der einzige Wert, der sich somit über die Zeit ändert, ist die Sonneneinstrahlung. Sie gibt an, wie hoch die Intensität des Solaren Spektrums ist, welches auf der Erde auftritt. Die Werte für die Sonneneinstrahlung können von vielen Wetterdienstanbietern abgerufen werden und sollen eine Auflösung von 15 Minuten haben. Da jedoch die Einstrahlungswerte durch die Wetterdienste auf horizontale Ebenen ausgegeben werden, müssen diese auf die jeweilige Neigung der PV-Anlage umgerechnet werden. Diese wird bei Dachanlagen durch die Neigung des Daches bestimmt, welche zumeist bei 30 $[^\circ]$ liegt. Die Umrechnung in die Einstrahlung auf eine geneigte Fläche ist jedoch nicht ganz trivial und beschreibt den komplexesten Teil der PV-Anlagen-Klasse. Um die Umrechnung durchzuführen, muss die Einstrahlung zunächst in ihre Bestandteile aufgeteilt werden. Die horizontale Einstrahlung lässt sich dabei in die Einstrahlungsanteile der Direktstrahlung G_b, der diffusen Einstrahlung G_d und der reflektierenden Strahlung G_r auf die horizontale Ebene aufteilen. Daher ist die Berechnung dieser Anteile auch der erste Schritt. Danach können die einzelnen Strahlungsanteile auf die geneigte Fläche berechnet und wieder zusammengesetzt werden.

Die Formeln für die Umrechnung wurden in der Implementierung verwendet und werden im folgenden Abschnitt 6.8.4 beschrieben. Das Ergebnis der Leistungsberechnung der PV-Anlage wird eine Leistungskurve sein, die alle 96 Werte für die 15 Minutenintervalle eines Tages beinhaltet. Jede unterschiedliche PV-Anlage wird eine eigene Leistungskurve berechnen. Die Leistungskurve stellt gleichzeitig die einzige Flexibilität der einzelnen PV-Anlagen dar. Dies ist damit zu Begründen, dass die PV-Anlagen von der Einstrahlung abhängig sind und immer nur dann elektrische Energie erzeugen können, wenn Sonneneinstrahlungen vorhanden sind. Da weiterhin im Projekt als Anforderung angegeben wurde, dass die PV-Anlagen all ihre Leistung verkaufen sollen, gibt es nur eine mögliche Flexibilität, die damit der Leistungskurve entspricht.

6.8.4. Implementierung Simulationsmodell PV-Anlage

Die PV-Anlagenmodellen werden durch die `PV` Klasse implementiert, welche im Flexibilitätenmodul integriert ist. Das Anlagenmodell für die Photovoltaikanlagen wurde auf Basis des Entwurfskapitels entwickelt und berechnet die tägliche Leistungskurve somit über die Formel 2:

$$P[W] := G[W/m^2] \cdot \eta \cdot A[m^2]$$

Formel 2: Leistungsberechnung bei einer PV-Anlage

Im Entwurfsabschnitt des Simulationsmodells der PV-Anlage (6.8.3) wurde weiterhin erläutert, dass für die Leistungsberechnung die solare Einstrahlung auf die geneigte Fläche berechnet werden muss. Für die Implementierung wurde deshalb für jeden Berechnungsschritt eine eigene Funktion in der PV-

Name der Variable	Beschreibung der Variable [Einheit]	Formel
doY	Tag des Jahres	siehe Formel 3
declination	Deklination [Radiant]	siehe Formel 4
timeEquation	Zeitgleichung	siehe Formel 5
hourAngle	Stundenwinkel [Winkelgrad]	siehe Formel 6
solarAltitude	Sonnenhöhe [Winkelgrad]	siehe Formel 7
azimuth	Azimuth [Radiant]	siehe Formel 8
arrivalAngle	Einfallswinkel [Winkelgrad]	siehe Formel 9
zenithAngle	Zenithwinkel [Winkelgrad]	siehe Formel 10
referenceRadiation	Referenzstrahlung [W/m^2]	siehe Formel 11
clarityIndex	Klarheitsindex	siehe Formel 12

Tabelle 4: Darstellung aller wichtigen Variablen zur Berechnung der geneigten Sonneneinstrahlung

Klasse erzeugt. Dazu wurden die folgenden zehn Werte genutzt, die für die Berechnungen relevant sind.

Neben den Parametern wurde für jede PV-Anlage eine txt-Datei erstellt, welche deren Stammdaten enthält. Die Stammdaten werden über den *PVTechnicalDataLoader* in das Modul eingelesen. Unter anderem sind hier auch der Wirkungsgrad und die Modulfläche enthalten. Damit diese aus der horizontalen Einstrahlung auf die geneigte Fläche umgerechnet werden kann, wurden zunächst die Einstrahlungswerte, des zu betrachtenden Tages, über das Datenbankmodul aus der Datenbank ausgelesen und an das Flexibilitätenmodul geschickt. Die gegebenen Werte für 60 Minuten Intervalle mit der Einheit J/m^2 werden in 15 Minuten Intervalle und W/m^2 umgerechnet. Im Folgenden soll die Betrachtung aller Berechnungsschritte erfolgen. Dazu wurde aus Übersichtsgründen der Text in zwei Unterabschnitte gegliedert. Der Erste Abschnitt stellt alle notwendigen Nebenrechnungen dar, während der zweite Abschnitt die Aufteilung auf die einzelnen Strahlungsanteile und die Umrechnung auf die geneigte Fläche erläutert.

Nebenberechnungen Zuerst erfolgt die Umrechnung des Datums in Tage seit Beginn des Jahres. Der Tag des Jahres (doY) wird mit Hilfe der `Calendar`-Klasse in Java implementiert (der 1. Januar entspricht dabei 1).

$$\text{doY} := \text{calendar.doY}(\text{date})$$

Formel 3: Berechnung: Tag des Jahres

wobei

calendar	Funktion in Java, um aus einem Datum den Tag des Jahres zu errechnen
date	Betrachtetes Datum

Anschließend wird der Wert in die Formel 4 aufgenommen, um die *Deklination* zu berechnen. Die Deklination beschreibt den Winkel zwischen Äquatorebene und der Sonne

$$\text{declination} := -23,45 \cdot \frac{10 + \text{doY}}{365}$$

Formel 4: Berechnung: Deklination

wobei

-23,45°	Beschreibt die maximale Neigung der Erdachse

Bei jeder Formel die mit Grad oder Radianten rechnet ist es wichtig, dass diese in Radianten konvertiert werden. In der Formel selbst wird nicht angegeben, ob es sich um Werte im Grad- oder Radiantenbereich handelt. Damit dies jedoch eindeutig ist, wurde für die Implementierung die Einheitenklassen *arcDegree* und *radiant* erstellt. Dadurch wird gewährleistet, dass die Kosinus- und Sinusfunktion angewandt werden kann. Die Deklination liegt dabei zwischen $-23,5°$ und $23,5°$, welches im Laufe des Jahres schwankt. Die Formel betrachtet dabei aber keine Schaltjahre, weshalb die 365 als fester Wert implementiert wurde. Nachdem die Deklination berechnet wurde, kann die Abweichung der Sonnenzeit von der Lokalzeit in Minuten über die Zeitgleichung (Formel 5) berechnet werden (feste Werte sind durch den Entwickler der Formel festgelegt worden) [Jor12]:

$$\text{timeEquation} := 60 \cdot (-0,171 \cdot \sin(0,0337 \cdot \text{doY} + 0,465) - 0,1299 \cdot \sin(0,01787 \cdot \text{doY} - 0,168)$$

Formel 5: Zeitgleichung

Mit der Sonnenzeit kann der Stundenwinkel kalkuliert werden, welcher den Winkelabstand der Sonne zur Südausrichtung in Grad angibt. Typischerweise entspricht dies ungefähr $15\,°/\text{h}$ [Mol08].

$$\text{hourAngle} := 15 \cdot \frac{\text{hoD} + \text{moD}}{60} - \left(\frac{15 - \text{longitude}}{15} - 12 + \frac{\text{timeEquation}}{60} \right)$$

Formel 6: Stundenwinkel

wobei

hoD	Stunde des Tages
moD	Minute des Tages
longitude	Standortlängengrad
60	Steht für die Umwandlung in Minuten
15	Wird verwendet, um nur jede 15 Minuten einen neuen Wert zu erhalten
12	Steht für den Zeitpunkt 12 Uhr

Das Besondere an Formel 6 ist, dass zusätzlich zur Nutzung des Standortlängengrades der jeweiligen Anlage, die Genauigkeit hier auf alle 15-Minuten-Intervalle des Tages erhöht wird. Der Standortlängengrad wird durch den `PVTechnicalDataLoader` ausgelesen, welcher im VPP-Modul implementiert wurde 6.4.1. Mit der Deklination und dem Stundenwinkel ist es nun möglich die Sonnenhöhe zu berechnen. Diese gibt den Winkel zwischen Sonnenmittelpunkt und dem Horizont des Beobachters an und liegt zwischen $-90°$ am Anfang sowie Ende des Tages, und $60°$ zur Mitte des Tages. Wobei es sich bei $-90°$ um den minimalen Wert im Winter und $60°$ den maximalen Wert im Sommer handelt. Berechnet wird die Sonnenhöhe [Jor12] über die Formel 7:

$$\begin{aligned} \text{solarAltitude} := \arcsin\big(& \sin(\text{latitude}) \cdot \sin(\text{declination}) \\ & + \cos(\text{latitude}) \cdot \cos(\text{declination}) \cdot \cos(\text{hourAngle}) \big) \end{aligned}$$

Formel 7: Sonnenhöhe

wobei

latitude	Standortbreitengrad

Im Gegensatz zu der Berechnung des Stundenwinkels wird für die Sonnenhöhe der Standortbreitengrad der Anlage benötigt. Auch dieser ist in den Anlagenstammdaten enthalten. Für die Berechnung des Azimuths, welcher den Winkel zwischen geografischer Südrichtung und dem Vertikalkreis durch den Sonnenmittelpunkt angibt, werden die vorher berechnete Deklination, Sonnenhöhe, Zeitgleichung sowie der Längen- und Breitengrad benötigt. Da die eigentliche Berechnung nach [Gie15] Werte bis 360° zulässt, der Azimuth für die weiteren Berechnungen jedoch zwischen $-180°$ und $+180°$ liegen muss, wurde die Berechnung durch eine Annahme erweitert, sodass eine entsprechende Verschiebung vorgenommen wird. Der Azimuth beschreibt dabei den Verlauf des Horiziontalwinkels mit dem die Himmelsrichtung bestimmt werden kann. Normalerweise wird Nord mit 0° und Süd mit 180° des Winkels definiert. Für die Formel jedoch wurde dies durch den Entwickler verändert, da auch bei den vorherigen Berechnungen negative Werte verwendet werden können und so spätere Rechnungen keine sinnvollen Ergebnisse liefern würden. Um die entsprechenden Werte zu erhalten, wurden für die Berechnung zwei Fälle aufgestellt. Hierbei mussten Annahmen getroffen werden, damit die Werte im angegebenen Bereich liegen. In beiden Fällen wird die Formel aus der Quelle [Gie15] verwendet. Der Unterschied liegt darin, dass allen Werten entweder $-180°$ oder 180° abgezogen werden. Dies muss durchgeführt werden, da sonst die Ergebnisse über 180° liegen. Beim ersten Fall wird der Azimuth zunächst mit $-180°$ angenommen und der Wert der eigentlichen Berechnung hinzu addiert. Dieser Fall gilt in den Stunden zwischen 0 und bis 12 Uhr. Danach wird der Fall zwei verwendet, indem von 180° der Wert der Berechnung abgezogen wird. Dadurch kann zu jedem Zeitpunk gewährleistet werden, dass sich der Wert im festgelegten Rahmen befindet. Da für die Azimuthberechnung der Stundenwinkel verwendet wird, werden auch die Azimuthwerte in einem Array aus 96-Werten gespeichert. Das bedeutet, dass bei der Implementierung jeweils über eine Schleife durch alle Werte iteriert werden muss. Die Formel des Azimuths sieht deshalb wie folgt aus und wurde gleichermaßen in *Powder* umgesetzt.

$$\begin{aligned}
\text{numerator} &:= -\sin(\text{latitude}) \cdot \sin(\text{solarAltitude}) - \sin(\text{declination}) \\
\text{denominator} &:= \cos(\text{latitude}) \cdot \sin(\arccos(\sin(\text{solarAltitude}))) \\
a &:= \frac{\text{numerator}}{\text{denominator}} \\
\text{azimuth} &:= \begin{cases} -180 + \arccos(a) & \text{falls } time \leq 12:00 \\ 180 - \arccos(a) & \text{sonst} \end{cases}
\end{aligned}$$

Formel 8: Azimuth

wobei

latitude	Standortbreitengrad
a	Wird als Variable verwendet, um die Berechnung aufzuspalten

Als nächstes wurde der Einfallswinkel (arrivalAngle) berechnet. Dabei handelt sich um den Winkel zwischen einfallendem Strahl und dem Normalvektor der Empfangsebene. Der Einfallswinkel kann Werte zwischen 0° und 90° annehmen. Hier wird ebenfalls eine Annahme getroffen, um nur Werte innerhalb des Wertebereichs zu erhalten. Es wurde neben der Formel eine If-Abfrage implementiert, in der jedes Ergebnis des Einfallswinkels, welcher größer als 90° ist, von 180° abgezogen wird. Die allgemeine Formel für den Einfallswinkel [Jor12] lautet:

$$
\begin{aligned}
\text{term1} &:= \sin(\text{declination}) \cdot \sin(\text{latitude}) \cdot \cos(\text{tiltAngle}) \\
\text{term2} &:= \sin(\text{declination}) \cdot \cos(\text{latitude}) \cdot \sin(\text{tiltAngle}) \cdot \cos(\text{azimuth}) \\
\text{term3} &:= \cos(\text{declination}) \cdot \cos(\text{latitude}) \cdot \cos(\text{tiltAngle}) \cdot \cos(\text{hourAngles}) \\
\text{term4} &:= \cos(\text{declination}) \cdot \sin(\text{latitude}) \cdot \sin(\text{tiltAngle}) \cdot \cos(\text{hourAngles}) \cdot \cos(\text{azimuth}) \\
\text{term5} &:= \cos(\text{declination}) \cdot \sin(\text{tiltAngle}) \cdot \sin(\text{azimuth}) \cdot \sin(\text{hourAngles}) \\
\text{arrivalAngleD} &:= \text{term1} - \text{term2} + \text{term3} + \text{term4} + \text{term5} \\
\text{arrivalAngle} &:= \arccos(\text{arrivalAngleD})
\end{aligned}
$$

Formel 9: Einfallswinkel

wobei

latitude	Standortbreitengrad
tiltAngle	Neigungswinkel der Anlage

Damit der Einfallswinkel berechnet werden kann, muss der Neigungswinkel der Anlage (tiltAngle) aus den Stammdaten mit ausgelesen werden. Dies geschieht mit dem `PVTechnicalDataLoader`.

Die letzte benötigte Nebenrechnung für die Umrechnung der horizontalen Strahlung in die Strahlung auf die geneigte Fläche ist die Berechnung des Zenitwinkels. Dieser gibt den Winkel zwischen einfallendem Strahl und dem Normalvektor der Horizontalen an. Auch dieser Winkel soll Werte zwischen 90° und 0° annehmen. Da auch hier bei ersten Berechnungen mit korrekter Formel [SH13] nicht verwendbare Werte erzeugt wurden, musste auch hier die Annahme getroffen werden, dass die Werte zwischen $-180°$ und 180° liegen. Wegen des Auftretens der nicht verwendbaren Werte in Zeiten, in denen die Sonnenhöhe dem Wert 0 entspricht und die Sonne dementsprechend noch nicht sichtbar ist, wurde hier eine Kondition angenommen, welche das Problem umgeht und passende Werte einsetzt. Diese überprüft den Wert der Sonnenhöhe. Solange die Sonnenhöhe dem Wert 0 oder kleiner entspricht, wird der Zenitwinkel als 90° angenommen. Auch der andere Teil der recherchierten Formel für die restlichen größeren Sonnenhöhe-Werte musste ersetzt werden. Deshalb wurde aus der Beziehung zwischen Sonnenhöhe und Zenit eine Formel abgeleitet. Für die Berechnung des Zenitwinkels gilt bei einer Sonnenhöhe größer 0 also:

$$
\text{zenithAngle} := \begin{cases} 90° & \text{falls solarAltitude} \leq 0 \\ 90° - \text{solarAltitude} & \text{sonst} \end{cases}
$$

Formel 10: Zenitwinkel

Mit der Berechnung des Zenitwinkels wurden alle wichtigen Nebenrechnungen für die Umrechnung implementiert und ausgeführt. Im folgenden Abschnitt wird die geneigte Gesamteinstrahlung über die vorliegenden Teilstrahlungen berechnet. Auch in diesem Fall wird mit Arrays gerechnet, die 96 Werte enthalten. Wie bereits angedeutet, wurden bei jeder Berechnung die Einheitenklassen verwendet, damit es nicht zu Fehlern bei der Umrechnung der Einheiten kommt.

Horizontale und geneigte Einstrahlung Die erste Teilstrahlung, die berechnet werden soll, ist die Diffusstrahlung. Sie beschreibt die Strahlung, welche von der Gesamtstrahlung durch Hindernisse reflektiert, absorbiert, gebrochen oder gestreut wird. Um diese zu berechnen, wird der Klarheitsindex benötigt. Dieser gibt das Verhältnis zwischen terrestrischer und extraterrestrischer Einstrahlung auf eine horizontale Fläche an. Um den Klarheitsindex zu kalkulieren, wird zunächst die Referenzstrahlung benötigt, welche die extraterrestrische Solareinstrahlung am äußeren Rand der Erdatmosphäre senkrecht zur Einstrahlungsrichtung angibt (Solarkonstante). Berechnet werden kann dies über die Formel 11 [Jor12]:

$$\text{referenceRadiation} := \text{solarConstant} \cdot \left(1 + 0{,}033 \cdot \cos\left(\frac{360}{365} \cdot \text{doY}\right)\right)$$

Formel 11: Referenzstrahlung

wobei

solarConstant	Langjährige gemittelte extraterrestrische Sonnenbestrahlungsstärke für den mittlerem Abstand zwischen Erde und Sonne, ohne den Einfluss der Atmosphäre senkrecht zur Strahlrichtung auf die Erde 1367 W/m^2
365	Anzahl der Tage in einem Jahr
fixe Werte	Sind durch den Entwickler der Formel festgelegt worden

Nach der Berechnung der Solarkonstante erfolgt die Berechnung des Klarheitsindex. Dieser gibt, wie zuvor erläutert, das Verhältnis zwischen extraterrestrischer und terrestrischer Strahlung an und muss noch berechnet werden. Bei der terrestrischen Strahlung handelt es sich um die Strahlungswerte, die durch die Wetterdienste ausgegeben werden. Wie bereits beschrieben, wurde diese für den betrachteten Tag über die interne Schnittstelle RMI aus dem Wettermodul an die PV-Klasse weitergegeben. Die Implementierung zur Berechnung des Klarheitsindex erfolgt über folgende Formel 12 [Jor12]:

$$\text{clarityIndex} := \frac{\text{globalSunshineCurve}}{\text{referenceRadiation}}$$

Formel 12: Klarheitsindex

wobei

globalSunshineCurve	Ausgelesene Sonneneinstrahlung des Betrachtungstages W/m^2

Mit dem Klarheitsindex wurde im Weiteren die Diffusstrahlung berechnet. Bei der Berechnung wurde wieder auf eine Fallunterscheidung gesetzt. Der Zusammenhang zwischen Klarheitsindex und Diffusstrahlung besteht folgendermaßen: Je größer der Klarheitsindex, desto kleiner ist der Anteil der

Diffusstrahlung an der Gesamtstrahlung. Es wurden bei der Implementierung nach [Jor12] drei Fälle integriert.

$$\text{diffuseRadiation} := \begin{cases} (1 - 0,249 \cdot \text{clarityIndex}) \cdot \text{globalSunshineCruve} & \text{falls clarityIndex} \leq 0,35 \\ 0,177 \cdot \text{globalSunshineCruve} & \text{falls clarityIndex} > 0,75 \\ (1,557 - 1,85 \cdot \text{clarityIndex}) \cdot \text{globalSunshineCruve} & \text{sonst} \end{cases}$$

Formel 13: Diffusstrahlung

wobei

globalSunshineCurve	Ausgelesene Sonneneinstrahlung des Betrachtungstages W/m^2
fixe Werte	Sind durch den Entwickler der Formel festgelegt worden

Die entstandene Diffusstrahlung wurde in einem nächsten Schritt in der Implementierung auf die geneigte Fläche umgerechnet. Dafür wurde das Modell von Liu-Jordan [Jor12] verwendet. Dies besagt folgendes Verhältnis:

$$\text{inclindedDiffuseRadiation} := \text{diffuseRadiation} \cdot \frac{1 + \cos(\text{tiltAngle})}{2}$$

Formel 14: Diffusstrahlung auf die geneigte Fläche

wobei

diffuseRadiation	Diffuse Sonneneinstrahlung aus Formel 13
tiltAngle	Neigungswinkel der Anlage
fixe Werte	Sind durch den Entwickler der Formel festgelegt worden

Die zweite Teilstrahlung ist die Direktstrahlung. Sie beschreibt den Strahlungsanteil, der direkt auf die PV-Module trifft. Für die Berechnung der Direktstrahlung gibt es eine Vielzahl von Berechnungsmodellen. Beim einfachsten Modell wird die Abhängigkeit der direkter Strahlung von diffuser Strahlung verwendet, denn beide zusammen ergeben die globale Sonneneinstrahlung. Bei dieser Abhängigkeit wird die dritte Teilstrahlung der Reflextion nicht betrachtet, da diese erst aus den anderen beiden hervorgeht. Die Direktstrahlung kann daher wie folgt ermittelt werden [Drü12]:

$$\text{directRadiation} := \text{globalSunshineCurve} - \text{diffuseRadiation}$$

Formel 15: Direktstrahlung

wobei

diffuseRadiation	Diffuse Sonneneinstrahlung aus Formel 13
globalSunshineCurve	Ausgelesene Sonneneinstrahlung des Betrachtungstages W/m^2

Über die Direktstrahlung kann die geneigte Direktstrahlung berechnet werden. Zusätzlich zu den vorherigen Annahmen aus den Berechnungen des Einfallswinkels (Formel 9) und des Zenitwinkels

(Formel 10) werden hier weitere Annahmen getroffen. Dies ist in der folgenden Berechnungsformel [Jor12] ersichtlich und wurde in *Powder* umgesetzt:

$$\text{inclinedDirectRadiation} := \begin{cases} 0 & \text{falls zenithAngle} >= 90^\circ \\ \cos\left(\frac{\text{arrivalAngle}}{\text{zenithAngle}}\right) \cdot \text{directRadiation} & \text{sonst} \end{cases}$$

Formel 16: geneigte Direktstrahlung

wobei

directRadiation	Direkte Sonneneinstrahlung aus Formel 15

Die erste Annahme hier ist, dass wenn der Zenitwinkel $\geq 90^\circ$ ist, der Wert für die direkte Strahlung bei 0 liegt. Dies geschieht zu allen Zeitpunkten, in der die Sonne noch nicht aufgegangen oder bereits untergegangen ist. Die zweite Annahme ist die Abänderung der Formel für den Fall, wo die Sonne tatsächlich am Himmel steht. Diese verwendet das Verhältnis des Cosinus des Einfallswinkels zum Cosinus des Zenitwinkels [Jor12]. Durch die Umrechnungen beim Zenit- und Einfallswinkel musste auch hier eine Umstellung vorgenommen werden. Infolgedessen wird der Cosinus aus dem Bruch herausgezogen.

Die Dritte und letzte Teilstrahlung ist die Reflektionsstrahlung. Diese betrachtet, wie der Name schon besagt, die reflektierte Strahlung vom Boden. Der Unterschied in der Berechnung zu den anderen beiden Strahlungsanteilen ist, dass die geneigte Reflektionstrahlung direkt, also ohne die Reflektionsstrahlung auf die horizontale Fläche, erfolgen kann [Jor12].

$$\text{inclinedReflectionRadiation} := \text{globalSunshineCurve} \cdot \frac{1 - \cos(\text{tiltAngle})}{2} \cdot \text{reflectionDegree}$$

Formel 17: Geneigte Reflektionsstrahlung

wobei

tiltAngle	Neigungswinkel der Anlage
globalSunshineCurve	Ausgelesene Sonneneinstrahlung des Betrachtungstages in W/m^2
reflectionDegree	Der Reflektionsgrad beschreibt den spezifischen Reflektionsfaktor des Untergrundes
fixe Werte	Sind durch den Entwickler der Formel festgelegt worden

Der benötigte Reflektionsgrad befindet sich in den Stammdatendateien, sodass dieser auch über den Loader ausgelesen wird. Dabei besitzen gut reflektierende Untergründe, wie Schnee, einen hohen Wert, was wiederum eine erhöhte Reflektionsstrahlung bedeutet.

Nach der Berechnung der geneigten Reflektionsstrahlung kann die Leistungskurve erzeugt werden. Dazu wird zunächst durch eine Addition aller Teilstrahlungen die Gesamtstrahlung auf die geneigte Fläche erstellt [Jor12].

$$\text{powerCurve} = \text{inclinedDiffuseRadiation} + \text{inclinedDirectRadiation} + \text{inclinedReflectionRadiation}$$

Formel 18: Leistungskurve

wobei

inclinedDiffuseRadiation	Geneigte Diffusstrahlung aus Formel 14
inclinedDirectRadiation	Geneigte Direktstrahlung aus Formel 16
inclinedReflectionRadiation	Geneigte Reflektionsstrahlung aus Formel 17

Wie auch die anderen Arrays in dieser Klasse, besteht die Liste aus Einstrahlungswerten auf die geneigte Fläche aus 96 Werten, welche die Einheit W/m^2 besitzen. Diese wird in einem nächsten Schritt in der Berechnung der Leistung 1 verwendet, um die Leistungskurve zu erzeugen. Die fertige Leistungskurve beschreibt, wie bereits im Entwurfskapitel 6.8.3 beschrieben, die einzige Flexibilität der PV-Anlagen. Unterschiedliche PV-Anlagen können im Flexibilitätenmodul über das Einlesen unterschiedlicher Stammdaten aus den entsprechenden txt-Dateien instanziiert werden. Die Leistungskurve jeder einzelnen PV-Anlage wird nach der Erstellung im Anlagenmodell dem Flexibilitätenmodul bereitgestellt. Neben dieser wird auch eine Kostenfunktion, welche die Kosten für das Produzieren von Energie angibt, übergeben. Diese wird im Abschnitt 6.8.7 genauer betrachtet. Durch die Umsetzung des PV-Modells wird auch die spezifische Anforderung L-01 erfüllt.

6.8.5. Entwurf Simulationsmodell BHKW

In diesem Kapitel werden Flexibilitäten durch Anlagenmodelle, wie z. B. BHKWs, beschrieben. BHKWs besitzen ein hohes Maß an Flexibilität in Bezug auf ihre Leistungsverteilung über den Tag. Durch einen Fahrplan (Schedule) lassen sich konkrete Tagesverläufe der elektrischen und thermischen Leistung darstellen. Um die Flexibilität der Anlage abzuschätzen, werden mehrere unterschiedliche Fahrpläne der gleichen Anlage erzeugt und als Flexibilitäten repräsentiert.

Bei der Erstellung von Fahrplänen müssen einige Aspekte beachtet werden. Zum einen muss die Leistung durch äquidistante Zeitpunkte repräsentiert werden, wie beispielsweise 96 Zeitpunkte pro Tag mit einer Dauer von je 15 Minuten. Im weiteren Verlauf wird von einem *Step* anstatt von *15* Minuten-Zeitpunkten gesprochen. Ebenso muss beachtet werden, dass für jeden Step die technischen Einschränkungen (Constraints) der Anlage eingehalten und zudem auch die Wärmeanforderungen des zum BHKW gehörigen Haushalts gedeckt werden. Im Rahmen der Recherche zu BHKWs haben sich vier wichtige Constraints herausgestellt. Das erste Constraint überprüft die noch zur Verfügung stehende Aufnahmefähigkeit der thermischen Leistung in einem Pufferspeicher. Einschränkungen durch die minimale Leistungsgrenze der Anlage sowie durch den Wechsel von Betriebszuständen stellen zwei weitere Constraints dar. Im letzten Constraint wird der Schmiermittelvorrat der Anlage mit dem Verbrauch verglichen.

Nachdem die Überprüfung der Constraints eines Steps erfolgreich war, können die Zustandsänderungen der einzelnen Komponenten der Anlage erfolgen sowie die Kosten und Erlöse generiert werden. Dieser Vorgang wird für jeden Step eines Tages wiederholt, bis alle Steps durchgeführt wurden. Der relative Leistungsanteil für jeden Step kann auf unterschiedliche Weise generiert werden. Eine Möglichkeit ist die Erzeugung von Zufallswerten für den relativen Leistungsanteil, der dann durch die anlagen spezifische Maximalleistung in die produzierte Leistung umgerechnet werden kann. Als Gesamtergebnis wird ein Schedule erzeugt, der für jeden Step die erzeugte elektrische Leistung beinhaltet.

Bezeichnung	Einheit	Beschreibung
MinElectricalPower	KiloWatt	Gibt die minimale elektrische Leistung an, die von der Anlage bereitgestellt werden kann.
MaxElektricalPower	KiloWatt	Gibt die maximale elektrische Leistung an, die von der Anlage bereitgestellt werden kann.
ElectricToThermal	DecimalPercentage	Umrechnungsfaktor elektrische zu thermische Leistung
MinTotalEfficiency	DecimalPercentage	Gibt die minimale Gesamteffizienz der Anlage an
MaxTotalEfficiency	DecimalPercentage	Gibt die maximale Gesamteffizienz der Anlage an

Tabelle 5: Stammdaten eines BHKWs

6.8.6. Implementierung Simulationsmodell BHKW

Im Folgenden wird die Implementierung der BHKW-Anlagen genauer betrachtet, die sich aus dem Schema der Abbildung 31 ergibt. Die Anforderung L-07 wird durch diese Implementation erfüllt. Die notwendigen Berechnungen der Fahrpläne für BHKWs werden in mehreren Teilschritten beschrieben.

Im ersten Schritt werden die ausgewählten Anlagentypen instanziiert. Dazu werden die Stammdaten und der aktuelle Status der Anlagen ausgelesen. Dies erfolgt im VPP-Modul und wird hier nicht näher betrachtet. Die daraus erhaltenen Daten werden in geeignete Klassenobjekte gespeichert. Diese implementieren für die Stammdaten das Interface `ITechData` und für den Status die abstrakte Klasse `AStatus`. Konkrete Klassen für ein BHKW sind `CHPStatus` und `CHP_LPG_Techdata`. In der ersten Klasse werden Zustandsinformationen über Schmiermittel, Pufferspeicherstand und Betriebszustand gespeichert. Die zweite Klasse beinhaltet Informationen über die Stammdaten, die in der Tabelle 5 nachgelesen werden können.

Als nächsten Schritt müssen für die BHKW-Anlagen mehrere unterschiedliche Fahrpläne erstellt werden, da die Anlagen, abhängig vom internen Zustand, unterschiedlich gesteuert werden können. Um eine möglichst große Abdeckung aller Fahrplan-Kombinationen zu erreichen, muss eine repräsentative Menge an Fahrplänen erstellt werden. Diese Verwaltungsstruktur wird von den sogenannten *Sampler*-Klassen übernommen, von denen drei vorhanden sind. Der `Sampler`, der `TSampler` und der `IntervalSampler`. Alle Sampler-Klassen klonen für jeden Fahrplan den Zustand der Anlage und wählen jeweils einen Leistungswert für alle äquidistanten Zeitintervalle aus. Dieser Leistungswert wird in einem für die Anlage spezifischen Modell simuliert und dadurch der Nachfolgezustand berechnet. Die Auswahl der Leistungswerte unterscheidet die Sampler voneinander. Der `Sampler` berechnet einen zufälligen Leistungswert, der `TSampler` berechnet ebenfalls einen zufälligen Leistungswert, lässt aber mehrere Anlagen simultan in unterschiedlichen Threads laufen. Der `IntervalSampler` lässt sich vorher eine Menge gültiger Leistungswerte berechnen, von denen einer zufällig ausgewählt wird. Zusätzlich sorgt er für eine kürzere Berechnungszeit, da bei der Simulation

mit rein zufälligen Leistungswerten häufig ein nicht gültiger Zustand erzielt wird und damit der Fahrplan verworfen werden muss.

Im letzten Schritt wird die Umsetzung der einzelnen Steps beschrieben. Dazu muss zunächst eine Fallunterscheidung vorgenommen werden. Im Fall des `Samplers` werden die Methoden `checkConstraints` und `Step` ausgeführt, wobei die Erste die nachfolgenden Constraints durchführt und die Zweite die Änderungen im Anlagenstatus durchführt. Identisch zu dem Ablauf verhält sich der `TSampler`. Im letzten Fall, in dem der `IntervalSampler` verwendet wird, berechnet die Methode `checkAllowedInterval` stattdessen ein Intervall mit gültigen Werten, bei dem die Constraints eingehalten werden.

Constraints Die Anlagen besitzen im Allgemeinen verschiedene Constraints , die bei der Berechnung der Flexibilitäten berücksichtigt werden müssen. Um das Überprüfen dieser Constraints so einfach wie möglich zu gestalten und die Constraints so zu implementieren, dass sie von verschiedenen Anlagentypen verwendet werden können, wurde für jedes Constraint eine eigene Klasse angelegt. Diese beinhaltet jeweils die statische Methode `checkConstraints` und `getAllowedInterval`. Dadurch ist es möglich, für jede Anlage eine spezifische Kombination von Constraints auszuwählen.

BufferStorageFillingLevelConstraint In der `BufferStorageFillingLevelConstraint`-Klasse wird für die erste Variante überprüft, ob genügend Kapazitäten für die erzeugte Wärmeleistung vorhanden ist. Für die Berechnung wird die minimale und maximale Temperatur des Pufferspeichers sowie der Folgezustand des Pufferspeichers nach Erzeugung der thermischen Leistung benötigt. In der Überprüfung wird festgestellt, ob sich der Folgezustand des Pufferspeichers in den Grenzen der minimalen und maximalen Temperatur bewegt. Ist dies der Fall, wird *true* zurückgeliefert. Der Folgezustand des Pufferspeichers wird in den Modellen der Anlage unter der Methode `calculateNewBufferStorageTemperature` berechnet. Für die Alternative wird diese Klasse nicht benötigt, da in der Methode `calculateAllowedBufferStorageIntervall` ein gültiges Intervall erzeugt wird.

calculateNewBufferStorageTemperature Diese Methode berechnet für einen Zeitschritt die neue Temperatur des Pufferspeichers. Dazu werden zwei Berechnungen vorgenommen. Die Erste berechnet die Abkühlung durch die Umgebungstemperatur und die Zweite den Zuwachs durch die thermische Leistung. Bei der Abkühlung lässt sich die neue Temperatur durch die Formel 19 beschreiben und erfüllt damit die Anforderung L-10.

$$
\begin{aligned}
AF &= \frac{V \cdot JinkWh}{SW \cdot KJinJ \cdot F \cdot 24H} \\
T1 &= T - AF \\
AK &= \frac{\ln(\frac{T1-Ut}{T-Ut})}{T1} \\
Tnew &= (T - Ut) \cdot e^{AK} + Ut
\end{aligned}
$$

Formel 19: Berechnung der Abkühlung eines Pufferspeichers

AF	Abkühlungsfaktor
V	Verlustwert des Pufferspeichers für 24 Stunden
$JinkWh$	Umrechnungsfaktor Joule in kWh
SW	Spezifische Wärmekapazität für Wasser c = 4,18
$KJinJ$	Umrechnung von KJ zu J, Faktor 1000
F	Füllstand des Pufferspeichers
$24H$	Konstante für 24 Stunden
$T1$	Unterschied zwischen aktueller Temperatur und dem Abkühlungsfaktor
T	Aktuelle Temperatur
AK	Abkühlungskoeffizient
$Tnew$	Neue Temperatur

Mit der Formel 20 lässt sich der Zuwachs bzw. die Abnahme der Wärme im Pufferspeicher berechnen. Dabei erhöht sich die Temperatur anhand der erzeugten thermischen Leistung und verringert sich durch den Haushaltsbedarf. Die Modellierung des Wärmeverbrauchs des Haushalts wird in Abschnitt 6.4.3 erläutert.

$$L = (P - C) \cdot kWhinKJ$$
$$Tnew = \frac{J}{SW \cdot F \cdot T}$$

Formel 20: Zuwachs / Abnahme der thermischen Leistung

L	Erzeugte/Verbrauchte Leistung in KiloJoule
P	Produzierte thermische Leistung
C	Verbrauchte thermische Leistung durch das Haushaltsprofil
$kWhinKJ$	kWh in KJ umrechnen, Faktor = 3600
$Tnew$	Neue Temperatur
SW	Spezifische Wärmekapazität für Wasser c = 4,18
F	Füllstand des Pufferspeichers
T	Aktuelle Temperatur

calculateAllowedBufferStorageIntervall Die Formeln aus der Methode `calculateNewBufferStorageTemperature` werden für die Berechnung eines gültigen Intervalls umgestellt. Dabei wird nicht die neue Temperatur im Pufferspeicher bei gegebener thermischer Leistung ermittelt, sondern es werden die thermischen Leistungsgrenzen, in denen die Grenzen des Pufferspeichers eingehalten werden, berechnet. Die untere Grenze kann durch die Formel 22 und die obere Grenze durch die Formel 21 ermittelt werden.

$$KJ = SW \cdot F \cdot (maxT - T)$$
$$OGinkWh = \frac{KJ}{kWhinKJ} - C$$

Formel 21: Berechnung der oberen Schranke

$$KJ = SW \cdot F \cdot (T - minT)$$
$$UGinkWh = \frac{KJ}{kWhinKJ} - C$$

Formel 22: Berechnung der unteren Schranke

L	Erzeugte/Verbrauchte Leistung in KiloJoule
C	Verbrauchte thermische Leistung durch das Haushaltsprofil
$maxT$	Maximale Temperatur im Pufferspeicher
$minT$	Minimale Temperatur im Pufferspeicher
$kWhinKJ$	kWh in KJ umrechnen, Faktor = 3600
$Tnew$	Neue Temperatur
SW	Spezifische Wärmekapazität für Wasser c = 4,18
F	Füllstand des Pufferspeichers
T	Aktuelle Temperatur
$OGinkWh$	Obere Grenze der erzeugbaren thermischen Leistung in kWh
$UGinkWh$	Untere Grenze der erzeugbaren thermischen Leistung in kWh

LubricantConstraint Die Klasse `LubricantConstraint` überprüft , ob genügend Schmiermittel in der Anlage vorhanden ist. Dazu wird der aktuelle Schmiermittelstand, der Schmiermittelverbrauch, die untere Grenze des Schmiermittels und die geplante Leistungsmenge benötigt. Die Implementierung unterscheidet dabei zwei Fälle:

- Die Anlage wird ausgeschaltet und es wird kein Schmiermittel verbraucht.
- Die Anlage läuft auf einer beliebigen Leistungsstufe und verbraucht Schmiermittel.

Ist sichergestellt, dass genügend Schmiermittel für die erwartete Leistungsstufe vorhanden ist, so wird *true* zurückgegeben. Die zweite Variante ermittelt die gültigen Leistungsstufen der Anlage, in denen genügend Schmiermittel für einen Betrieb vorhanden sind. Dazu wird ebenfalls das zuvor erwähnte Zustandswissen benötigt, jedoch wird die geplante Leistungsmenge nicht als Input erwartet. Zurückgegeben wird ein Objekt der Klasse `Interval`, welches die erlaubten Leistungsstufen beinhaltet.

OperatingStateConstraint Ebenso, wie in der Klasse `LubricantConstraint`, wird auch in der Klasse `OperatingStateConstraint` die Berechnung in zwei Varianten unterteilt. Die erste Variante überprüft, ob die gewünschte Leistungsstufe durch die Anlage umgesetzt werden kann. Folgende Zustände werden benötigt: Aktueller Betriebszustand, Aktiv- und Inaktivzeit, minimal benötigte Aktivzeit und minimal benötigte Inaktivzeit. Für jeden Zustand, aktiv oder inaktiv, wird überprüft, ob ein Wechsel in den jeweils anderen Zustand erfolgen darf. Dazu werden die Zeiten der jeweiligen Betriebszustände mit der minimal nötigen Zeit verglichen. Liegt kein Wechsel der Betriebszustände vor oder ist dieser möglich, wird *true* zurückgeliefert. Für die andere Variante werden, unabhängig von der geplanten Leistung, alle möglichen Leistungsmengen berechnet und als Objekt der Klasse

`Interval` zurückgegeben. Dazu werden alle unter der ersten Variante beschriebenen Informationen benötigt, mit Ausnahme der geplanten Leistungsmenge.

PerformanceLimitConstraint Die Klasse `PerformanceLimitConstraint` wird ebenfalls in zwei Varianten umgesetzt. Die erste Variante überprüft, ob die geplante Leistungsmenge mit den Anlagenspezifischen Angaben übereinstimmt. Dazu werden Informationen über die geplante Leistungsmenge und der minimalen Leistungsgrenze (ohne 0) benötigt. Bei der zweiten Variante wird ebenfalls ein Objekt der Klasse `Interval` zurückgeben. Dieses beinhaltet die minimale und maximale Leistungsgrenze der Anlage und zusätzlich die 0 als mögliche Leistungsstufen. Die 0 bedeutet, dass die Anlage ausgeschaltet ist.

6.8.7. Kosten- und Erlösberechnung

Dieses Kapitel beschäftigt sich mit der Kosten- und Erlösberechnung der Anlagen. Als Kosten werden nur Betriebskosten und keine Anschaffungskosten betrachtet, wodurch die Anforderungen L-02 und L-08 abgedeckt sind. Hingegen wurde die Anforderung L-11 verworfen, da die Verzögerungen der Leistung bei BHKWs mit LPG sehr gering sind und dadurch vernachlässigt werden können. Dies wurde nachträglich als abgeänderte Anforderung mit den Projektbetreuern besprochen. Die Berechnungen werden je Anlagenart getrennt betrachtet und im nachfolgendem beschrieben.

Kostenberechnung von PV-Anlagen Bei der Kostenberechnung von PV-Anlagen werden im Rahmen der Projektgruppe lediglich die fixen und variablen Betriebskosten, die in einem Jahr anfallen, berücksichtigt. Die Investitionskosten für die einzelne Anlage sollen dabei nicht betrachtet werden (s. Anforderungen 4.3). Die anfallenden Kosten für die PV-Anlagen sind dabei im Gegensatz zu BHKW gering, da nur Wartungskosten für die Module und Wechselrichter anfallen. Da bei den PV-Anlagen keine beweglichen Teile enthalten sind, fallen keine Kosten für Verschleiß sowie ebenfalls keine Kosten für Kraftstoffe an. Die Wartungskosten können dabei für ein Jahr über eine Faustformel berechnet werden und liegen bei 35€ pro installierte kWp [BSfWuM]. Die berechneten Kosten müssen zudem entsprechend der Anforderungen der Projektgruppe auf jeweils einen Tag berechnet werden. Für die Implementierung wurde im Flexibilitätenmodul aus den jlPVTechdata der jeweils ausgewählten PV-Anlage die maximale kWp ausgelesen. Durch die maximale kWp und die nach der Faustformel berechneten Kosten von 35€ werden die jährlichen Kosten berechnet. Im selben Schritt werden daraus die täglichen Kosten abgeleitet. Die Kosten werden nur auf einen Tag heruntergerechnet, da die Kostenberechnung der PV-Anlagen nicht leistungsabhängig ist. Dies liegt daran, dass die Kosten lediglich fixe Kosten sind und die Berechnung dadurch einfacher durchzuführen ist. Daher kann an sonnenarmen und wolkenreichen Tagen der Fall eintreten, dass die PV-Anlage keine Gewinne erzielt, sondern Verluste einfährt. Dafür ist der Gewinn an sonnenreichen Tagen im Vergleich zu den Kosten hoch. Diese Kosten werden mit den Erlösen aus der Zuschlagsberechnung zusammenaddiert und das Ergebnis wird zusammen mit den Flexibilität übergeben.

Zusatzerlöse von PV-Anlagen Wie bereits in Abschnitt 6.8.4 erläutert, werden für die Einsatzplanoptimierung neben der erzeugten Energie auch die Kosten der einzelnen Fahrpläne benötigt, da-

mit der Gewinn optimiert werden kann. Neben den Kosten der Fahrplane und den Erlösen von der Börse wird auch auf die nach dem EEG mögliche Direktvermarktung zurückgegriffen. Diese erlaubt zunächst das Handeln an der Börse für EEG-Anlagen kann zu weiteren Gewinnen kommen. Diese möglichen Gewinne werden in der Projektgruppe als Zusatzerlöse für PV-Anlagen definiert. Die Direktvermarktung darf verwendet werden, da es sich bei den PV-Anlagen per Definition um EEG-Anlagen handelt [BMW](EEG Stand 2014). Bei der Direktvermarktung handelt es sich um den direkten Verkauf von Strom aus Erneuerbaren Energien an direkte Kunden oder an einer Strombörse. Die Direktvermarktung kann, nach dem Wegfall des Grünstromprivilegs EGG 2014 [BMW], in zwei unterschiedlichen Ausprägungen in Anspruch genommen werden.

Die erste Ausprägung ist die Direktvermarktung nach dem *Marktprämienmodell*. Dabei handelt es sich um ein durch das Bundesministerium eingesetztes Instrument, mit dem die Marktintegration der Erneuerbaren Energie-Anlagen gefördert werden soll [NEX15]. Neben dem Erlös aus dem Verkauf des Stroms an der Börse erhält der Anlagenbetreiber die Marktprämie von dem jeweiligen Übertragungsnetzbetreiber ausgezahlt. Die Marktprämie übernimmt somit das bisherige EEG-Vergütungsmodell für die Anlagen, die in der Direktvermarktung eingesetzt werden. Für die Berechnung der Marktprämie wird ein anlagenspezifischer Referenzmarktwert und die EEG-Einspeisevergütung verwendet, die im Folgenden noch genauer erläutert wird (s. Abbildung 32). Aus der Abbildung geht

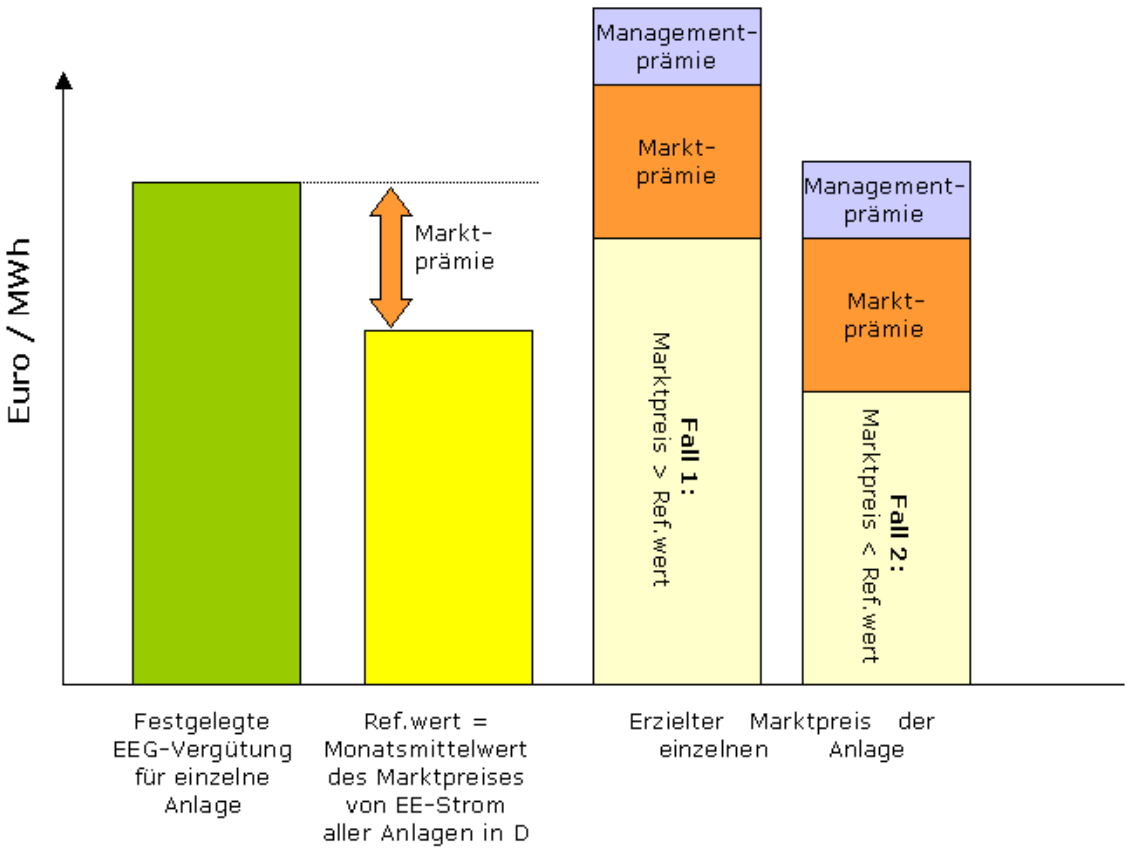


Abbildung 32: Marktprämienmodell [Pho15]

weiterhin hervor, dass eine Gewinneinbuße gegenüber der EEG-Vergütung entstehen würde, falls der

Referenzmarktwert größer ist als der Börsenpreis. Anders herum würden bei einem größeren Börsenpreis aber auch Mehrerlöse generiert werden. Dadurch besitzt das Marktprämienmodell gegenüber der festen EEG-Einspeisevergütung ein Risiko, aber auch eine Chance auf Mehrerlös. Bei der zweiten Ausprägung handelt es sich um die sogennante *sonstige Direktvermarktung*. Bei dieser erhält der Anlagenbetreiber lediglich die Erlöse von der Börse, daher wird dieser Ansatz kaum verwendet [NEX15].

Auch die Projektgruppe hat sich aus Gründen des höheren Erlöses für die Direktvermarktung nach dem Marktprämienmodell entschieden. Für die Berechnung der Marktprämie wird folgende Formel verwendet:

$$\text{Marktpraemie} = fixe\ Einspeiseverguetung - Referenzmarktwert \tag{5}$$

$$\text{Referenzmarktwert} = Energietraegerspezifischer\ Marktwert - Managementpraemie \tag{6}$$

Formel 23: Berechnung der Marktprämie [NEX15]

fixe Einspeisevergütung	Betrag, den die Anlage über die EEG-Einspeisevergütung erhält
Referenzmarktwert	(bzw. Monatsmarktwert) Wert der von der fixe Einspeisevergütung abgezogen wird um die Marktprämie zu erhalten
Energieträgerspezifischer Marktwert	Durchschnittliche Stundenpreise am Spotmarkt, welche mit den energieträgerspezifischen Faktoren verrechnet werden
Managementprämie	Prämie für den Mehraufwand und das Mehrrisiko der Direktvermarktung

Zur Implementierung der Marktprämie für die Zuschläge werden also die fixe Einspeisevergütung der jeweiligen PV-Anlagen, die Managementprämie und der energieträgersepezifische Marktwert benötigt, um den Referenzmarktwert und die Marktprämie berechnen zu können. Da jedoch der Referenzmarktwert aus der Quelle [NT16] entnommen werden kann, werden die Managementprämie und der energieträgersepezifische Marktwert nicht benötigt. Der Referenzmarktwert ist dabei in der Quelle für jeden Monat ab dem Jahr 2012 vorhanden, welche für das Projekt in der Datenbank hinterlegt sind. Die fixe Einspeisevergütung kann aus der Quelle [BSW] ausgelesen werden. Dazu wird das Installationsdatum, welches für alle Anlagen auf den 1.1.2016 gesetzt wurde, und die installierte Leistung in kWp der Anlagen benötigt. In der implementierten Funktion `getEEGReward` wird unter Betrachtung der installierten Leistung dann über eine Abfrage die EEG-Einspeisevergütung der Anlage ausgewählt. Daneben ist zu beachten, dass Anlagen über eine installierte Leistung von 10 MW nur die ersten 10 MW über das EEG vergütet bekommen und somit auch nur für diese Leistung einen Anspruch an die Marktprämie besitzen. Alle weitere installierte Leistung erhält bei Verkauf nur den Börsenpreis. In der Funktion `calculateRewards` wird neben dem Aufruf der `EEGReward` auch der Referenzmarktwert für den betrachteten Tag ausgelesen, wo-

für der `PVReferenceMarketValueDataProvider` verwendet wird. Mit den beiden Werten kann die Marktprämie für die einzelnen Anlagen berechnet werden.

In die Flexibilitäten werden dann die Ergebnisse aus der Funktion `PVOperatingCostsAndRewards`, welche die Kosten für die Erzeugung und die Zuschläge für die Vermarktung enthält, aufgenommen.

Kostenberechnung von BHKW-Anlagen Zur Berechnung der Kosten für das Betreiben von BHKW-Anlagen werden verschiedene Faktoren betrachtet. Ein Großteil der Kosten entsteht durch den verbrauchten Kraftstoff sowie durch die Wartung der Anlagen. Hingegen sind die Kosten für den Schmiermittelverbrauch, besonders bei kleinen Anlagen, in den Wartungskosten inbegriffen. Ebenfalls ergeben sich bei kleinen Anlagen keine Abgaben durch die erzeugten Emissionen.

Der Verbrauchte Kraftstoff wird anteilig an der erzeugten elektrischen Leistung und der dazugehörigen Verlustleistung bemessen und erfüllt damit die Anforderung L-09. Die Kosten für den Kraftstoff errechnen sich aus den durchschnittlichen Preisen in ganz Deutschland und können für Flüssiggas aus [Bun] entnommen werden. In der Formel 24 wird genauer auf die Umsetzung eingegangen. Die Wartungskosten lassen sich einfach auf die erzeugte elektrische Leistung umlegen und haben einen festen Wert pro erzeugte mWh, die in [Hei] nachgelesen werden kann. Die bereits erwähnten Kosten durch den Schmiermittelverbrauch und den Emissionsausstoß können für sehr große Anlagen berechnet werden, in dem die dafür bereitgestellten Methoden genutzt werden.

$$TP = maxEP \cdot EtT \tag{7}$$

$$totalP = maxEP + TP \tag{8}$$

$$L = \frac{totalP}{maxTotalE} - totalP \tag{9}$$

$$EL = \frac{L}{1 + EtT} \tag{10}$$

$$GP = (EL + maxEP) \cdot P \tag{11}$$

$$C = FC \cdot GP \cdot SA \tag{12}$$

Formel 24: Kostenberechnung für den Verbrauchten Kraftstoff

TP	Thermische Leistung in KW
$maxEP$	Maximale elektrische Leistung durch die Anlage
EtT	Faktor um von elektrischer Leistung auf die thermische Leistung zu schließen
$totalP$	Gesamtleistung
L	Verlustleistung durch die Anlage
$maxTotalE$	Maximale Effizienz der Anlage
EL	Anteil der Verlustleistung, die der elektrischen Leistung zugeordnet werden kann
GP	Erzeugter Leistungsanteil
P	Aktuelle Leistungsstärke der Anlage zu einem `Step`
C	Kosten, die durch den Verbrauch in einem `Step` entstehen
FC	Kosten für den Kraftstoff pro erzeugte mWh
SA	Zeitlicher Anteil an einer Stunde

Zusatzerlöse von BHKW-Anlagen Ebenso, wie die Kosten für BHKW-Anlagen, werden auch Erlöse für die jeweiligen Fahrpläne bestimmt. Diese richten sich nach dem Gesetz für KWK-Anlagen und unterteilen sich grob in den KWK-Zuschlag und dem Netzentgelt durch den Netzbetreiber [fWuA].

Das Netzentgelt schwankt zwischen 0,4 und 1,5 Cent/kWh und richtet sich nach dem jeweiligen Standort. Zur Vereinfachung wurde ein Mittelwert von 1 Cent/kWh angenommen. Die Berechnung für den KWK-Zuschlag wird in der Formel 25 konkretisiert.

$$K := \frac{€}{mWh} \cdot \begin{cases} 80 & \text{falls } maxEP < 50kWh \\ \frac{(50 \cdot 80 + (maxEP - 50) \cdot 60}{maxEP} & \text{falls } 50kWh < maxEP < 100kWh \\ \frac{(50 \cdot 80 + 50 \cdot 60 + (maxEP - 100) \cdot 50}{maxEP} & \text{falls } 100kWh < maxEP < 250kWh \\ \frac{(50 \cdot 80 + 50 \cdot 60 + 150 \cdot 50 + (maxEP - 250) \cdot 44}{maxEP} & \text{falls } 250kWh < maxEP < 2000kWh \\ \frac{(50 \cdot 80 + 50 \cdot 60 + 150 \cdot 50 + 1750 \cdot 44}{maxEP} & \text{falls } maxEP > 2000kWh \end{cases}$$

Formel 25: KWK-Zuschlagsberechnung

K	KWK-Zuschlagsberechnung
$maxEP$	Maximale elektrische Leistung durch die Anlage

6.9. Optimierungsmodul

Das Optimierungsmodul ist die zentrale Komponente von *Powder*. Dort laufen die Informationen aus dem Marktmodul und dem Flexibilitätenmodul zusammen. Begrenzt durch die Flexibilitäten der Anlagen wird in dem Modul eine Kombination von Marktprodukten, inklusive der Kapazitäten und Preise für die Produkte, so gewählt, dass die Einnahmen maximiert werden. Zugleich wird für jede Anlage ein Fahrplan ausgewählt und somit einen Einsatzplan erstellt. Diese beiden Erzeugnisse sind aufeinander abgestimmt und werden an das VK-Modul übermittelt.

Nachfolgend wird zunächst die grundlegende Problemstellung (s. Abschnitt 6.9.1) ausführlich erläutert, worauf eine detaillierte Betrachtung der beiden einzelnen Optimierungsprobleme (s. Abschnitt 6.9.2 und 6.9.3) und eines kombinierten Ansatzes (s. Abschnitt 6.9.4) erfolgt. Im Anschluss daran wird auf die Implementierung (s. Abschnitt 6.9.5) und die Metaoptimierung (s. Abschnitt 6.9.6) eingegangen. Abschließend wird die Angebotsberechnung (s. Abschnitt 6.9.7) erläutert und Abweichungen zu den Anforderungen (s. Abschnitt 6.9.8) begründet.

6.9.1. Grundlagen

Im Rahmen des Projekts müssen zwei Optimierungsprobleme gelöst werden. Zum einen soll ein Produktportfolio so zusammengestellt werden, dass mit den gegebenen Anlagen und auf Grundlage der Wetter- und Marktprognose der Gewinn maximiert wird (Produktportfoliooptimierung, PPO). Zum anderen muss eine Auswahl der Anlagenfahrpläne getroffen werden, mit der die für das Produktportfolio erforderliche Energie möglichst kostenoptimal (Einsatzplanoptimierung, EPO) erzeugt wird. Es erfolgt also eine Optimierung der Produktportfolios und des Einsatzplans des virtuellen Kraftwerks.

Als Grundlage für die Gesamtoptimierung werden die Marktprognose (siehe Kapitel 6.7) und eine Menge von Flexibilitäten (s. Kapitel 6.8.1) benötigt. Die Flexibilitäten stellen einen zufälligen Ausschnitt von möglichen Fahrplänen der Anlagen dar.

Abbildung 33 veranschaulicht die Problemstellung des Optimierungsprozesses. Die Flexibilitäten der Anlagen sowie die Marktprognose stellen die Basis dar. Ziel der Optimierung ist, das Produktportfolio und den Einsatzplan so zu wählen, dass der Gewinn maximiert wird.

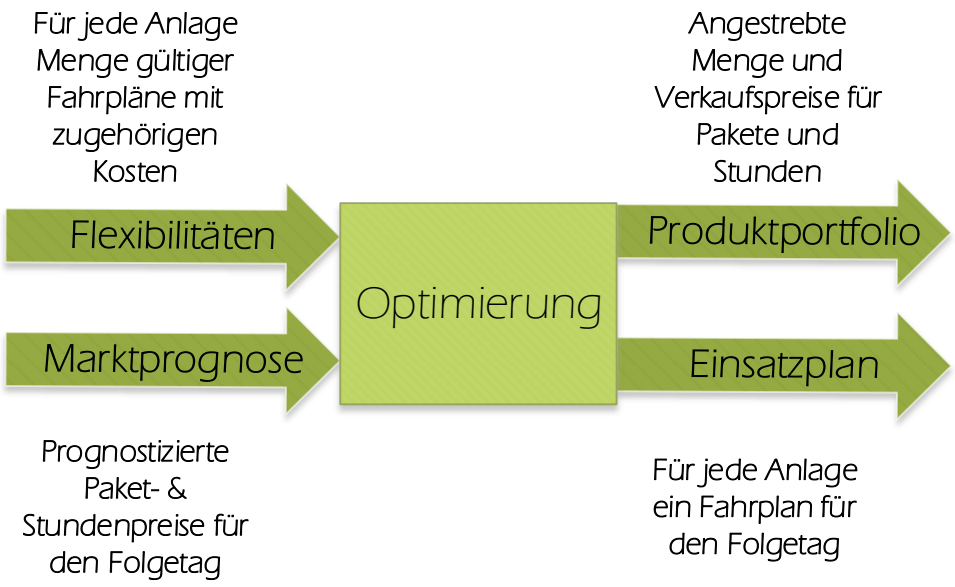


Abbildung 33: Problemstellung der Optimierung

Für die formale Beschreibung der Zielfunktion werden folgende Variablen benötigt:

i	ein bestimmtes Marktprodukt, wobei $i \in \{0,1,\ldots,m\}$
m	Anzahl der Marktprodukte
$Einnahmen_i$	Einnahmen, die durch den Verkauf des Produkts i zum gewählten Preis und in der gewählten Kapazität entstehen
a	eine bestimmte Anlage, wobei $a \in \{0,1,\ldots,n\}$
n	Anzahl der Anlagen
$Kosten_a$	Kosten, die durch die Ausführung des gewählten Fahrplans für Anlage a entstehen
Zusatzerlöse_a	Zusatzerlöse (durch gesetzliche Zulagen), die durch die Ausführung des gewählten Fahrplans für Anlage a entstehen
$P_e(t)$	durch den Einsatzplan erzeugte Energie zum Zeitpunkt t
$P_v(t)$	verkaufte Energie zum Zeitpunkt t

Für das Gesamtoptimierungsproblem ergibt sich die Zielfunktion:

$$Max\left(\sum_{i=0}^{m} Einnahmen_i - \sum_{a=0}^{n} (Kosten_a - \text{Zusatzerlöse}_a)\right) \tag{13}$$

Die Einnahmen der gewählten Produkte sollen maximiert werden, während die durch den Einsatzplan erzeugten Kosten, abzüglich der durch staatliche Zulagen generierten Zusatzerlöse, minimiert werden sollen. Durch diese Zielfunktion wird die Leistungsanforderung L-17 erfüllt, welche festlegt, dass der Gewinn maximiert werden soll. Darüber hinaus müssen die durch die Anlagen erzeugte Energie und die am Markt verkaufte Energie aufeinander abgestimmt werden. Wird mehr Energie verkauft,

als durch das virtuelle Kraftwerk erzeugt wird, werden Vertragsstrafen fällig. Laut Qualitätsanforderung Q-02 muss das System keine Vertragsstrafen berücksichtigen, es ist aber dennoch sinnvoll, das Anfallen von Strafzahlungen zu vermeiden. Wird mehr Energie erzeugt, als am Markt verkauft wurde, kann die überschüssige Energie nicht gehandelt werden, wodurch keine Einnahmen entstehen aber dennoch Produktionskosten anfallen. Die fluktuierende Einspeisung der PV-Anlagen ermöglicht keine zu 100% genaue Vorhersage der erzeugten Energie, wodurch eine exakte Abstimmung von Erzeugung und Verbrauch nicht möglich ist. Da die Zahlung von Vertragsstrafen zu vermeiden ist, wird festgelegt, dass zu jedem Zeitpunkt mindestens so viel Energie produziert werden muss, wie verkauft wurde:

$$P_v(t) \leq P_e(t)$$

Die beiden Leistungsverläufe $P_v(t)$ und $P_e(t)$ sind Ergebnisse und zugleich Vorgaben für PPO und EPO. Diese beiden Optimierungen sind wechselseitig miteinander verbunden, was die Gesamtoptimierung zu einem komplexen Problem macht. Beide Optimierungen benötigen die Vorgabe eines Leistungsverlaufs. Für die Bestimmung geeigneter Marktprodukte und deren Kapazitäten muss die verfügbare Energiemenge bekannt sein. Zugleich wird ein Einsatzplan normalerweise so optimiert, dass die kumulierte Leistung der Anlagen einem vorgegebenen Leistungsverlauf entspricht. Daher ist die Produktportfoliooptimierung abhängig von einem festen Einsatzplan, wohingegen die Einsatzplanoptimierung ein vorgegebenes Produktportfolio benötigt.

Diese wechselseitige Abhängigkeit kann aufgelöst werden, indem eine der beiden Optimierungen mit einer Heuristik ausgeführt wird. Durch die Nutzung der Heuristik wird kein vorgegebener Leistungsverlauf mehr benötigt – stattdessen werden zufällige Lösungen erzeugt, deren Güte anschließend bewertet wird. Für die Bewertung der Lösung ist das Gesamtoptimierungsergebnis, bestehend aus Produktportfolio und Einsatzplan, nötig. Deshalb erfolgt eine Aufteilung in eine äußere und eine innere Optimierung. Die äußere Optimierung nutzt eine Heuristik für die Generierung von zufälligen Lösungen. Anschließend wird die innere Optimierung verwendet, um das korrespondierende Optimierungsergebnis (Produktportfolio oder Einsatzplan) zu ermitteln. Daraufhin kann die Güte der Gesamtlösung, also der erzielte Gewinn berechnet werden.

Nachfolgend werden die beiden Optimierungsprobleme einzeln betrachtet, um festzustellen, welche Optimierung sich besser als äußere Optimierung und welche sich besser als innere Optimierung eignet.

6.9.2. Optimierung des Produktportfolios

Die Produktportfoliooptimierung hat das Ziel, auf Basis der Marktprognose das Produktportfolio so zusammenzustellen, dass die Einnahmen möglichst groß werden. Dabei darf nur soviel Energie verkauft werden, wie durch die Anlagen des virtuellen Kraftwerks erzeugt werden kann.

Das Produktportfolio beinhaltet, zu welchen Zeiten wie viel Energie geliefert werden muss. Dabei kann aus verschiedenen Produkten des European Power Exchange Spotmarket gewählt werden. Diese beinhalten sowohl Einzelstunden als auch einige standardisierte Blockprodukte, die über mehrere Stunden gehen. Zudem besteht die Möglichkeit eigene Produkte zu konfigurieren, was aufgrund der

unendlichen Möglichkeiten hier nicht betrachtet wird. Weitere Details zu EPEX-Spot-Produkten können im Anhang im Seminarband Anhang G nachgelesen werden. Diese Auswahl der Produkte und daraus folgende Zusammenstellung des Produktportfolios erfüllen die Anforderungen PPO-PUC4 und PPO-PUC5.

Für jedes Stundenintervall muss eine Auswahl der Produkte und deren Kapazität, also der Menge an Energie in Megawatt (MW) sowie des Verkaufspreises getroffen werden. Dabei ist zu beachten, dass pro Produkt eine Mindestgröße von 0,1 MW erreicht werden muss und eine Maximalgröße von 600 MW nicht überschritten werden darf. Zudem kann die Kapazität der Produkte nur in Schritten von 0,1 MW inkrementiert werden. Die Angaben zu Marktregeln sind aus [Eur16] entnommen.

Blockprodukte erstrecken sich über mehrere Stunden, wobei ein Block nur in seiner Gesamtheit oder überhaupt nicht verkauft werden kann. Der Preis pro Megawattstunde (MWh) des Blockes errechnet sich aus dem Mittelwert der Preise der Einzelstunden, welche dem Block angehören. Innerhalb eines Blockgebots können zu den verschiedenen Stunden unterschiedliche Energiemengen verkauft werden. Ein „Blockverkaufsgebot wird jedoch nicht ausgeführt, wenn sein Preislimit höher als die volumengewichteten durchschnittlichen markträumenden Preise für die Stundenkontrakte ist, auf die er sich bezieht"[Eur16]. Dieser Sachverhalt wird nachfolgend anhand eines Beispiels näher erläutert. In Abbildung 34 werden drei Einzelstunden und ein Blockprodukt, welches diese Stunden beinhaltet dargestellt.

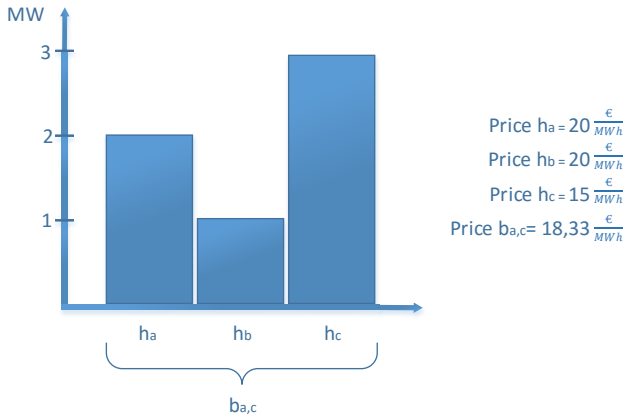


Abbildung 34: Beispielhafte Darstellung der Ausführungsbedingung für Blockprodukte

Würde die zur Verfügung stehende Energie in Stundenkontrakten verkauft, ergäbe dies einen volumengewichteten, durchschnittlichen und markträumenden Preis von

$$\frac{2 \cdot 20 \frac{€}{MWh} + 1 \cdot 20 \frac{€}{MWh} + 3 \cdot 15 \frac{€}{MWh}}{6} = 17,5 \frac{€}{MWh}$$

Das Preislimit des Blocks $b_{a,c}$ beträgt jedoch $18,33 \frac{€}{MWh}$ (Mittelwert der Stundenpreise). Somit würde der Block mit der abgebildeten Volumenverteilung nicht ausgeführt werden. Um diesen Umstand zu vermeiden wird für die Optimierung festgelegt, dass während der gesamten Dauer eines Blocks eine gleichbleibende Energiemenge auszuwählen ist. Auf diese Weise wird eine Gleichheit des Preislimits und des volumengewichteten, durchschnittlichen und markträumenden Preises garantiert.

Da der Preis pro Megawattstunde für ein Blockprodukt aus dem Mittelwert der zugehörigen Einzelstunden berechnet wird, besteht kein rein finanzieller Anreiz Blockprodukte anzubieten. Der Mehrwert entsteht durch die *alles-oder-nichts*-Eigenschaft der Blockprodukte. Werden nur Einzelstunden angeboten, kann die Situation auftreten, dass einige Stunden verkauft werden und andere nicht, wodurch Lücken im Leistungsverlauf, der durch das Produktportfolio vorgegebenen wird, entstehen. Ein solcher lückenhafter Leistungsverlauf kann meist aufgrund technischer Einschränkungen nicht durch die Anlagen des virtuellen Kraftwerks realisiert werden. Dies hat zur Folge, dass in den Stunden, welche nicht veräußert werden konnten, dennoch Energie durch die Anlagen erzeugt wird. Es fallen Produktionskosten an, wobei zugleich keine Einnahmen am Day-Ahead-Handel erzeugt werden. Die einzige Vermarktungsmöglichkeit besteht darin, diese Energie im Intraday-Handel zu verkaufen. Dies ist aber mit weiterem Aufwand verbunden und birgt wiederum das Risiko, dass Angebote nicht angenommen werden.

Damit die Abwägung zwischen Einzelstunden und Blöcken in der Optimierung sinnvoll getroffen werden kann, muss die Abnahmewahrscheinlichkeit in Abhängigkeit vom gewählten Angebotspreis miteinbezogen werden. In diesem Projekt standen die erforderlichen Informationen für die Berechnung der Abnahmewahrscheinlichkeit nicht zur Verfügung. Dieser Aspekt kann bei den nachfolgend erläuterten Lösungsvarianten ergänzt werden. An entsprechender Stelle wird näher darauf eingegangen.

Die Energie, die pro Stunde verkauft werden kann, ist, wie bereits in Kapitel 6.9.1 erläutert wurde, nach oben hin durch den Einsatzplan beschränkt. Eine Auswahl von Objekten zu treffen, wobei der Gesamtwert maximiert werden soll und zugleich eine bestimmte (Gewichts-)Schranke nicht überschritten werden darf, weist starke Ähnlichkeiten mit der Klasse der Rucksackprobleme auf. Da diese Problemklasse ausgiebig erforscht wurde, bietet sie einen guten Ausgangspunkt für die theoretische Betrachtung des Problems. Aus diesem Grund werden im Folgenden zunächst das klassische Rucksackproblem beschrieben und anschließend eine Abbildung der PPO auf ein Rucksackproblem und somit eine Formalisierung vorgenommen.

Rucksackproblem Das Rucksackproblem ist ein NP-vollständiges, kombinatorisches Optimierungsproblem. Es kann formal wie folgt beschrieben werden:

Gegeben ist eine Menge von n Objekten. Jedes Objekt i besitzt einen Nutzen p_i und ein Gewicht w_i. Außerdem existiert ein „Rucksack“ mit einer bestimmten Kapazität c. Das Ziel ist, den „Rucksack“ so mit den Objekten zu füllen, dass der Nutzen maximal wird, aber zugleich nicht die Kapazität überschritten wird [KPP04]. Dies kann mit folgendem mathematischen Modell beschrieben werden:

$$\max : \left(\sum_{i=1}^{n} p_i x_i \right) \tag{14}$$

Unter der Bedingung:

$$\sum_{i=1}^{n} w_i x_i \leq c \tag{15}$$

$$x_i \in 0,1, \quad i = 1 \cdots n \tag{16}$$

Wobei x_i angibt, ob das Element i in den Rucksack gepackt wurde, oder nicht. Die Lösung des Problems wird also durch einen binären Vektor $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ dargestellt, der die Auswahl der Objekte widerspiegelt.

Abbildung des PPO auf das Rucksackproblem Das Problem der Produktportfoliooptimierung weist viele Parallelen zu dem klassischen Rucksackproblem auf. Nachfolgend wird davon ausgegangen, dass ein Einsatzplan vorliegt, für den das Portfolio optimiert werden soll. Wird für die Gesamtoptimierung das umgekehrte Vorgehen gewählt, so muss zu Beginn dennoch ein Referenzeinsatzplan erstellt werden, mit dem das Optimierungsverfahren beginnen kann. Für die Formalisierung der PPO werden folgende Variablen benötigt:

i	ein bestimmtes Marktprodukt, wobei $i \in \{0,1,\ldots,m\}$
m	Anzahl der Marktprodukte
$Preis_i$	Preis, für den Produkt i verkauft werden soll in $\frac{€}{MWh}$
v_i	Volumen des Produkts i pro Stunde in MW
d_i	Dauer des Produkts i in Stunden
$c(h)$	Erzeugte Leistung des Einsatzplans in der Stunde h in MW, mit $h \in \{0,1,\ldots,23\}$
$V(h)$	Menge von Produkten, die in der Stunde h verfügbar sind, mit $h \in \{0,1,\ldots,23\}$

Somit erfolgt eine Zuordnung der Variablen des Rucksackproblems auf die der PPO:

Rucksackproblem	**PPO**
Kapazität des Rucksacks c	c(t)
Gewicht des Objekts i: w_i	v_i
Wert des Objekts i: p_i	$Preis_i \cdot d_i$

Für die PPO kann also folgendes mathematisches Modell aufgestellt werden:
Zielfunktion:

$$\max \left(\sum_{i=0}^{m} Preis_i \cdot v_i \cdot d_i \right) \tag{17}$$

Diese Zielfunktion maximiert die Einnahmen durch das Produktportfolio, wodurch die Qualitätsanforderung Q-01 erfüllt wird. Zugleich müssen folgende Restriktionen eingehalten werden:

$$\sum_{v_i \in V(h)} v_i \leq c(h) \tag{18}$$

Es wird zu jedem Zeitpunkt t nicht mehr Leistung verkauft, als durch den Einsatzplan erzeugt wird.

$$v_i = 0 \quad \vee \quad 0,1MW \leq v_i \leq 600MW \tag{19}$$

Die Größe eines Produkts muss entweder 0 sein (Produkt nicht gewählt) oder innerhalb des zulässigen Wertebereichs liegen. Außerdem sind nur diskrete Werte in Schritten von 0,1 MW möglich.

Die Lösung des Problems wird also durch einen Vektor $V = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ dargestellt, der die Auswahl der Leistungsmengen der Objekte widerspiegelt.

Die PPO unterscheidet sich vor allem durch die zeitliche Dimension vom herkömmlichen Rucksackproblem. Diese Problemeigenschaft bewirkt eine Zunahme an Komplexität. Für jede Stunde des Tages muss das Problem erneut gelöst werden, da sich sowohl die verfügbaren Produkte, als auch die vorhandene Kapazität ändern. Einige Produkte dauern mehrere Stunden, wobei die Leistung gleich bleiben muss. Aus diesem Grund ergeben sich Abhängigkeiten zwischen den Stunden.

Wie zu Beginn in Abschnitt 6.9.2 erwähnt wurde, handelt es sich beim Rucksackproblem um ein NP-vollständiges Problem. Die Berechnungskomplexität dieser Klasse von Problemen erhöht sich durch ansteigende Variablenanzahl extrem. Daher werden oft Heuristiken zur Lösung verwendet. Im hier vorliegenden Fall ist die Eingabe jedoch begrenzt. Es werden 41 standardisierte Produkte am EPEX-Spot-Market gehandelt. Aus diesem Grund können auch mathematisch exakte Verfahren verwendet werden. Die Leistungsanforderung L-16 legt fest, dass das Produktportfolio in einem automatisierten Prozess zusammengestellt werden muss. Im Folgenden werden zwei Lösungsmöglichkeiten für die PPO erläutert.

Linear Programming-Optimierer „Das mathematische Teilgebiet der linearen Optimierung behandelt Probleme, in denen eine von n Veränderlichen abhängige lineare Funktion, die Zielfunktion, unter Einhaltung linearer Restriktionen, den Nebenbedingungen, zu minimieren oder maximieren ist. Dies kann als die Suche einer im Sinne der Zielfunktion optimalen Entscheidung unter Vorgabe gewisser Bedingungen interpretiert werden.“ [UD10]

Bei der in Abschnitt 6.9.2 vorgenommenen mathematischen Formulierung des Problems handelt es sich bereits um ein lineares Optimierungsproblem. Als solches bezeichnet man Aufgaben der Form

$$max/min \quad p^T x$$
$$u.d.B. \quad Ax \lesseqgtr c$$

wobei $x_i \cdots x_n$ die Entscheidungsvariablen sind, $p_i \cdots p_n$ die Nutzenkoeffizienten, $c_i \cdots c_n$ die Beschränkungen und A die Koeffizientenmatrix für Nebenbedingungen [UD10].

Diese Formalisierung weist große Ähnlichkeit mit dem bereits entwickelten Modell auf. Lediglich Constraint 18 muss in mehrere Ungleichungen unterteilt werden. Dieses relativ kleine Gleichungssystem kann mit bewährten Methoden der Linearen Optimierung, wie beispielsweise dem Simplex-Algorithmus, gelöst werden. Dabei liegt die durchschnittliche Laufzeitkomplexität in $O(n^2)$ [UD10].

In diesem Projekt wurde der *Linear Programming*-Optimierer (LP-Optimierer) für die Umsetzung des linearen Optimierungsproblems implementiert. Dabei wird der Linear-Programming-Solver Gurobi zur Lösung des Gleichungssystem verwendet. Das zuvor beschriebene Gleichungssystem kann mit geringem Aufwand in einem Gurobimodell inplementiert werden. Dazu werden Variablen und Terme angelegt und der Zielfunktion und den Constraints zugeordnet. Die Lösung wird darauf automatisch von Gurobi berechnet.

Sollen zu einem späteren Zeitpunkt die Abnahmewahrscheinlichkeiten der Produkte miteinbezogen werden, so muss lediglich die Zielfunktion angepasst werden. Die Faktoren $Preis_i \cdot d_i$ müssen durch einen Faktor ersetzt werden, welcher die Abnahmewahrscheinlichkeit in Abhängigkeit vom gewählten Preis beinhaltet.

Rekursiv-Intervall-Optimierer Zusätzlich zum linearen Optimierer wurde ein intervallbasiertes Lösungsverfahren entwickelt. Ziel dieses Rekursiv-Intervall-Optimierers (RI-Optimierer) ist wie bereits oben beschrieben die Suche nach einem Produktportfolio, welches möglichst große Einnahmen erzielt.

Durch den Einsatzplan ist die verfügbare Energiemenge E in stündlicher Auflösung gegeben. Es erfolgt eine sukzessive Aufteilung in rechteckige Teilbereiche, was in Abbildung 35 für drei Rekursionsschritte dargestellt wird. In jedem Rekursionsschritt wird das gegebene Intervall an der Stelle der geringsten Leistung unterteilt, wodurch neue Intervalle entstehen.

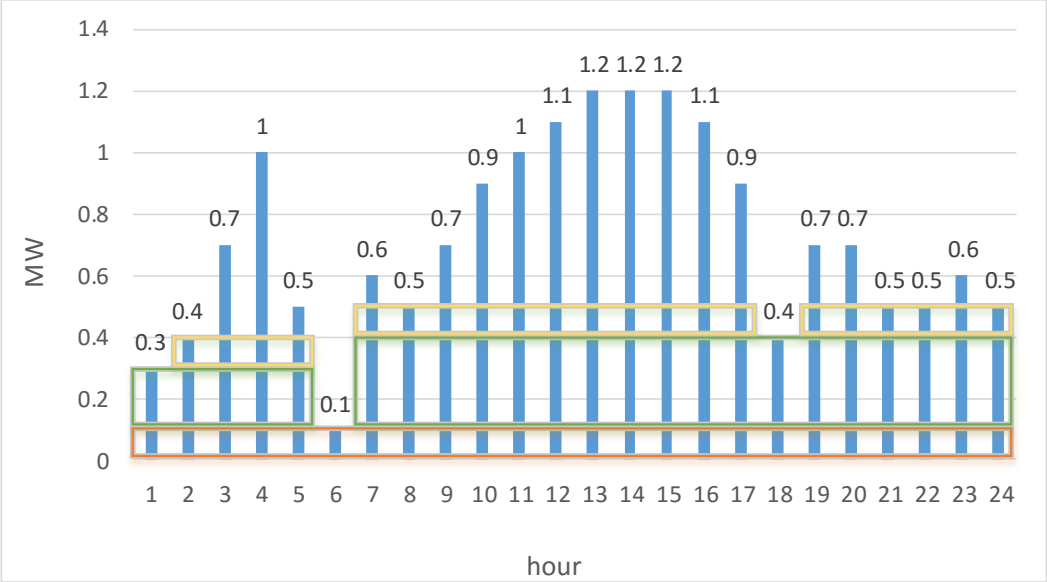


Abbildung 35: Beispielhafte Zerlgeung der Intervalle

Ein Intervall wird folgendermaßen definiert $I = (u, v, p)$, wobei u und v Stunden des Tages und p die geringste Leistung im Intervall (gegeben in MW) repräsentieren. Für jedes Intervall wird eine optimale Kombination von Produkten berechnet. Dabei wird ein ähnliches Verfahren wie beim *Dynamic Programming* verwendet.

Optimale Produktkombinationen werden nacheinander für Teilintervalle aufsteigender Größe bestimmt. Für jedes dieser Teilintervalle $I_t = [u, i]$ mit $u \leq i \leq v$, werden alle Marktprodukte ermittelt, die zu einem beliebigen Zeitpunkt in I starten und genau zum Zeitpunkt i enden. Aus diesen Produkten wird dasjenige ausgewählt, welches in Verbindung mit den in vorherigen Teilintervallen berechneten Produktkombinationen den besten Durchschnittspreis ergibt. Dadurch wird schrittweise eine optima-

le Kombination von Marktprodukten für I erstellt. In Abbildung 36 wird dieser Prozess anhand eines beliebigen Beispiels, mit $I = (a,d,p)$ und $d = a+3$ erläutert.

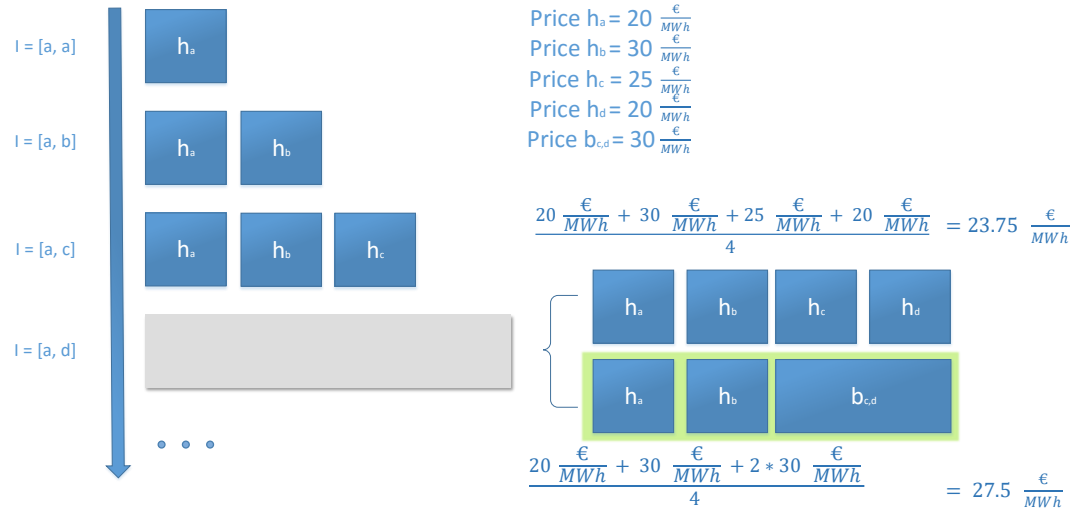


Abbildung 36: Suche des besten Preises pro Stufe des Intervalls

Im ersten Schritt wird das Teilintervall $I_t = [a,a]$ betrachtet. In dieser Zeitspanne von einer Stunde ist nur das Marktprodukt der Einzelstunde h_a verfügbar, welches als initiale Lösung für diesen Teilabschnitt verwendet wird. Im nächste Teilintervall $T_i = [a,b]$ endet nur das Marktprodukt der Einzelstunde h_b, welches mit dem vorherigen Teilintervall h_a zusammengeführt wird und die optimale Produktkombination (h_a,h_b) ergibt. Die gleiche Situation tritt bei $I = [a,c]$ auf und liefert als Ergebnis (h_a,h_b,h_c). In $I = [a,d]$ existieren zwei mögliche Marktprodukte, die zum Zeitpunkt h_d enden: Die Einzelstunde h_d und das Blockprodukt $b_{c,d}$. Daraus entstehen zwei Lösungsvorschläge: (h_a,h_b,h_c,h_d) und $(h_a,h_b,b_{c,d})$. Werden diese miteinander verglichen, ergibt sich ein höherer Durchschnittspreis durch den zweiten Lösungsvorschlag, sodass dieser als optimale Kombination ausgewählt wird. Anschließend wird allen Produkten in der optimalen Kombination die im Intervall verfügbare Energiemenge p zugewiesen. Nach dem Durchlauf aller Intervalle werden Energiemengen, welche in verschiedenen Intervallen dem selben Produkt zugeordnet wurden, aufsummiert, um die Gesamtmenge des Produktes festzulegen.

Im Nachfolgenden wird genau beschrieben, wie die Umsetzung implementiert wurde. Die Klasse `RecursiveIntervalPPOS` besitzt vier relevante Methoden. Die Methode `optimize` stellt das Grundgerüst der Klasse dar und ruft sich selber rekursiv auf. In dieser Methode werden zunächst Wert und Zeitpunkt der geringsten Leistung im gegebenen Intervall berechnet. Anhand dieser Informationen werden vorab einige Randfälle überprüft, wie beispielsweise ob die Zeitspanne weniger als eine Stunde beträgt oder ob die geringste Leistung unter 0.1 MW liegt. Treten diese Fälle nicht ein, dann wird die nächste Methode `findBestPriceForRangeHours` aufgerufen, in der wie zuvor beschrieben eine optimale Produktkombination gewählt wird.

Nachdem die Kombination des Intervalls bestimmt wurde, kann die Leistungsmenge p anschließend in der Methode `setEnergy` den Produkten zugeordnet. In der Methode `reduceEnergy` wird die verfügbare Gesamtleistung E um die bereits verteilte Leistungsmenge p reduziert: $E' = E - p$. Anschließend werden die Energiemengen identischer Produkte zusammengeführt und der nächste Rekursionsschritt wird eingeleitet.

Ähnlich wie bei der linearen Optimierung können zu einem späteren Zeitpunkt Abnahmewahrscheinlichkeiten der Produkte miteinbezogen werden, indem bei der Auswahl der Produktkombinationen nicht mehr der Durchschnittspreis als Vergleichskriterium genutzt wird, sondern ein Faktor der die Abnahmewahrscheinlichkeit in Abhängigkeit vom gewählten Preis beinhaltet.

Bei beiden vorgestellten Umsetzungsmöglichkeiten für die PPO werden die Einnahmen des Produktportfolios und somit die Güte berechnet. Dadurch wird die Anforderung PPO-PUC8 erfüllt.

6.9.3. Optimierung des Einsatzplans

Die Einsatzplanoptimierung wählt auf Grundlage der Flexibilitäten der Anlagen einen Einsatzplan aus. Der Einsatzplan legt fest, welche Fahrpläne aus der Menge aller realisierbaren Fahrpläne die Anlagen zu fahren haben. Dabei kann pro Anlage genau ein Fahrplan gewählt werden. Ein Fahrplan enthält Informationen über die zu erbringende elektrische Leistung in 15-minütiger Zeitauflösung.

Nachfolgend wird zunächst eine formale Betrachtung des Problems vorgenommen. Anschließend werden verschiedene heuristische Lösungsverfahren im Allgemeinen erläutert und an das spezifische Problem angepasst.

Formalisierung Durch die endliche Anzahl an verfügbaren Fahrplänen von Anlagen ist es möglich, die Einsatzplanoptimierung mittels eines linearen Gleichungssystems zu lösen. Dabei muss ein zu erreichender Leistungsverlauf vorgegeben sein. Für die Formalisierung werden folgende Variablen benötigt:

a	eine Anlage, wobei $a \in \{0, 1, \ldots, n\}$
n	Anzahl aller Anlagen des VKs
j	ein Fahrplan einer Anlage, wobei $j \in \{0, 1, \ldots, m_a\}$
m_a	Anzahl von Fahrplänen der Anlage a
f_{a_j}	Fahrplan j von Anlage a
x_{a_j}	boolescher Ausdruck, der angibt ob Fahrplan f_{a_j} gewählt wurde
$k(f_{a_j})$	Kosten für Fahrplan f_{a_j} in $\frac{€}{MWh}$
$z(f_{a_j})$	Zusatzerlöse für Fahrplan f_{a_j} durch staatliche Zulagen in $\frac{€}{MWh}$
$l_{h_q}(f_{a_j})$	Leistung von Fahrplan f_{a_j} in der Stunde h und dem Viertelstundenintervall q in MW
$t(h)$	Untergrenze der zu generierenden Leistung in Stunde h in MW

Das Ziel ist die Kosten abzüglich der Zusatzerlöse zu minimieren. Als Maximierungsproblem formuliert lautet dies folgendermaßen:

$$max\left(\sum_{a=0}^{n}\sum_{j=0}^{m_a}\left(z(f_{a_j}) - k(f_{a_j})\right)\cdot x_{a_j}\right) \tag{20}$$

$$t(h) \leq \sum_{a=0}^{n} \sum_{j=0}^{m_a} l_{h_q}(f_{a_j}) \cdot x_{a_j}, \quad \forall h \in \{0,\ldots,23\}, \quad \forall q \in \{0,\ldots,3\} \tag{21}$$

Es wird in jeder Stunde mindestens so viel Energie produziert, wie für die Einhaltung der Untergrenze benötigt wird. Dabei gilt die Untergrenze pro Stunde, während die Einsatzpläne viertelstündlich aufgelöst sind. Daher muss die Summe der erzeugten Energie durch alle gewählten Fahrpläne für jedes Viertelstundenintervall die momentan gültige stündliche Untergrenze überschreiten.

Durch ein weiteres Constraint wird sichergestellt, dass genau ein Fahrplan pro Anlage gewählt wird:

$$\text{Pro Anlage } a: \sum_{j=0}^{m_a} x_{a_j} = 1 \tag{22}$$

Dieses Gleichungssystem kann mit Linear-Programming-Solvern (wie Gurobi) mit geringem Aufwand implementiert und anschließend gelöst werden. Die Laufzeit wächst jedoch quadratisch in Abhängigkeit von der Anzahl an Variablen. Da unter anderem für jeden möglichen Fahrplan jeder Anlage eine Variable benötigt wird, führt dies entweder zu sehr langen Laufzeiten oder stark eingeschränkten Lösungsmöglichkeiten. Diese stark eingeschränkten Lösungsmöglichkeiten sind durch eine geringe Anzahl von möglichen Fahrplänen bedingt.

In Abschnitt 6.9.2 wurde gezeigt, dass die PPO im Gegensatz zur EPO gut durch exakte Berechnungsverfahren wie der linearen Optimierung oder dem selbst entwickelten Rekursiv-Intervall-Optimierer gelöst werden kann. Aus diesem Grund eignet sich die PPO als innerer Optimierer, wohingegen die EPO besser durch eine Heuristik gelöst werden kann und somit den äußeren Optimierer darstellt. Nachfolgend werden verschiedene Heuristiken, die für die EPO verwendet werden können, erläutert.

Evolutionärer Algorithmus Der Evolutionärer Algorithmus ist ein stochastisches, naturinspiriertes Optimierungsverfahren, welches das Prinzip SSurvival of the fittestümsetzt. Dabei wird mit einer Population aus einzelnen Individuen bzw. Lösungen des Problems gearbeitet. Diese verändert sich, indem durch Rekombination und Mutation neue Generationen erzeugt werden. Durch eine große Population und einen adäquaten Grad an Zufälligkeit kann ein Lösungsraum gut durchsucht werden.

Der Ablauf des Evolutionären Algorithmus wird in Abbildung 37 dargestellt. Zu Anfang wird eine Startpopulation P (zufällig) generiert und evaluiert. Danach beginnt der Generationenzyklus, in dem zuerst ρ Eltern gewählt werden (Selektion), aus welchen mit Rekombination ein neues Individuum x_i erzeugt wird. Die Auswahl der Eltern kann z. B. anhand ihrer Fitness oder durch ein Wettkampfverfahren ermittelt werden. Ein einfaches Rekombinationsverfahren ist der n-Punkt-Crossover. Bei diesem werden, analog zur biologischen Rekombination von DNS, die Codierungen der Eltern an mehreren Punkten aufgesplittet, getauscht und wieder zusammengefügt. Das neu geschaffene Individuum wird anschließend mutiert. Dazu wird in der Regel ein beliebiger Parameter zufällig verändert oder das gespeicherte Objekt auf Binärebene manipuliert. Je nach konkretem Anwendungsfall kann auch versucht werden, so zu mutieren, dass wieder eine gültige Lösung entsteht. Mit einer Fitnessfunktion wird das erstellte Individuum x_i bewertet und abschließend der neuen Generation P' hinzugefügt. Auf

diese Weise werden λ Nachkommen erzeugt. Bei der Selektion der neuen Generation ist zwischen der Plus- und der Komma-Selektion zu unterscheiden. Bei der Plus-Selektion $(\mu + \lambda)$ werden μ Elemente für die neue Generation aus der alten und der neuen Generation gewählt ($P_{neu} \subset P \cup P'$). Bei der Komma-Selektion (μ, λ) werden diese nur aus der neuen Generation ($P_{neu} \subseteq P'$) gewählt. Es werden solange neue Generationen berechnet, bis ein Terminierungskriterium erfüllt wird. Dieses kann z. B. durch eine feste Anzahl an Generationen, eine maximale Rechenzeit oder geringe Veränderung der Güte des besten Individuums definiert werden.

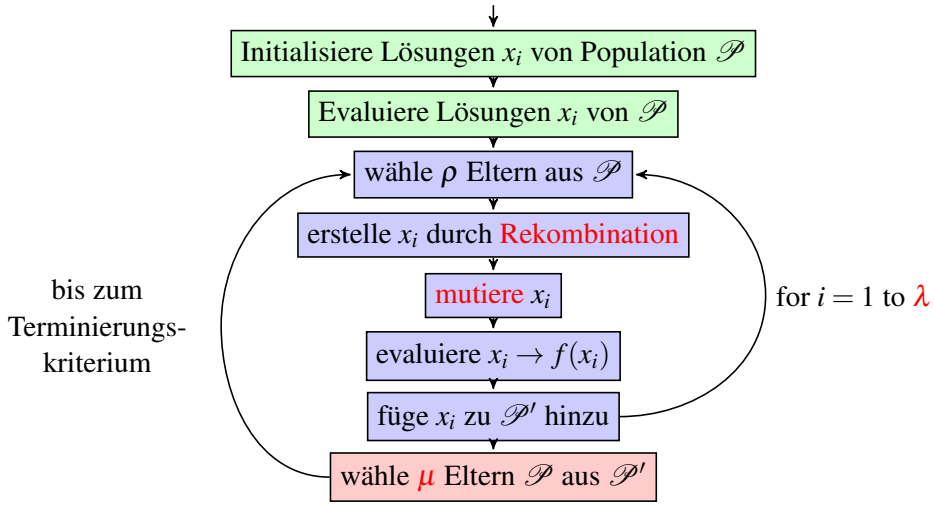


Abbildung 37: Ablauf des Evolutionären Algorithmus

Simulated Annealing-Optimierer Mit *Simulated Annealing* wurde das Optimierungsproblem des Einsatzplans heuristisch angegangen. Die Funktionsweise des Verfahrens ähnelt dem der Evolutionären Algorithmen. Hierbei gibt es eine aktuelle Temperatur T, die iterativ auf eine Mindesttemperatur T_{min} abgekühlt wird. In jeder Iteration wird heuristisch nach einem Optimum gesucht. Abbildung 38 beschreibt den Ablauf visuell. Es wird zunächst eine Lösung zufällig als aktuelle Lösung x ausgewählt. Iterativ wird x mutiert und es entsteht eine neue Lösung x'. Anschließend werden die beiden Lösungen x und x' miteinander verglichen. Besitzt die mutierte Lösung x' eine höhere Fitness, als die aktuelle Lösung x, so wird x' als neue Lösung gewählt, andernfalls wird die schlechtere Lösung x', nur mit einer bestimmten Wahrscheinlichkeit, als neue Lösung gewählt. Diese Wahrscheinlichkeit hängt von der aktuellen Temperatur ab. Je höher die Temperatur ist, desto wahrscheinlicher wird die schlechtere Lösung x' gewählt. So wird versucht, ein Stagnieren an lokalen Optima zu verhindern. Anschließend wird die Temperatur T abgesenkt. Solange die Temperatur die Mindesttemperatur T_{min} übersteigt, wird einer weitere Iteration eingeleitet. Andernfalls wird die aktuelle Lösung x als gefundenes Optimum zurückgeliefert.

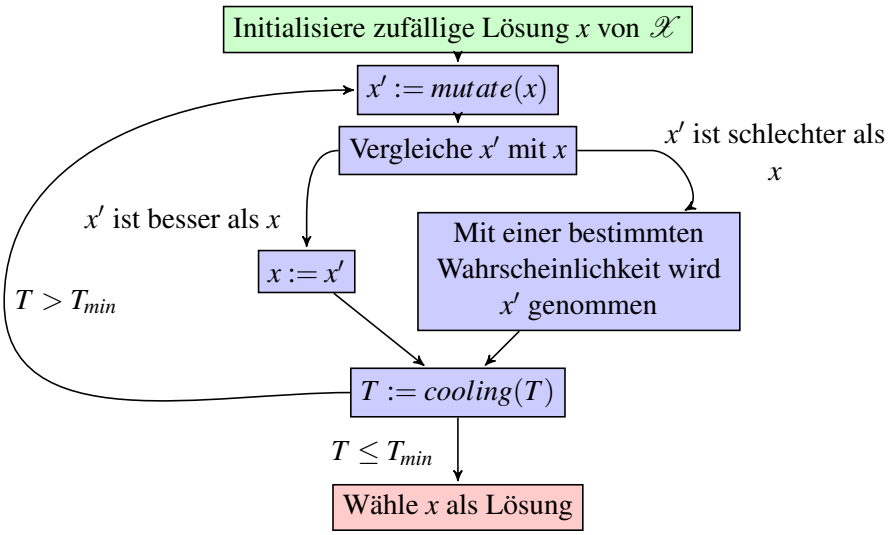


Abbildung 38: Ablauf beim Simulated Annealing

Im *Simulated Annealing*-Optimierer (SA-Optimierer) von *Powder* repräsentiert ein Individuum einen Einsatzplan. Zur Berechnung der Fitness eines Individuums wird ein passendes Produktportfolio benötigt. Zu diesem Zweck wird einer der in Kapitel 6.9.2 erläuterten Produktportfoliooptimierer verwendet. Die Fitness eines Individuums ist der Gewinn, welcher durch die Kombination des Einsatzplans und des Produktportfolios entsteht. Die Mutation eines Einsatzplans beinhaltet die Mutation jedes enthaltenen Fahrplans in Abhängigkeit von der Mutationsstärke σ. Dabei nimmt σ einen Wert zwischen 0 und 1 an. Ist $\sigma = 1$, wird in jedem Intervall eines Fahrplans die zu erzeugende prozentuale Leistung durch einen zufälligen Wert $\in [0 \cdots 1]$ ersetzt. Je kleiner σ ist, desto geringer ist der Anteil an Intervallen, die mutiert werden. Die Mutationstärke wird nach Rechenbergs 1/5-Regel im Laufe der Optimierung angepasst.

Die Mutation der Einsatzpläne beinhaltet die besondere Schwierigkeit, dass Fahrpläne die Restriktionen der Anlagen berücksichtigen müssen. Um die Einhaltung von Anlagenconstraints und die Optimierung zu entkoppeln, werden Dekodierer verwendet. Ein Dekodierer erhält einen beliebigen Fahrplan für eine Anlage als Eingabe und gibt einen gültigen Fahrplan zurück, welcher dem ursprünglichen möglichst ähnlich sein soll.

Die Funktionsweise des Dekodierers wird in Kapitel Abschnitt 6.9.3 erläutert. Da der SA-Optimierer zeitgesteuert ist, sinkt die Temperatur proportional zur verbliebenen Zeit, um den Suchraum in den letzten Optimierungsschritten lokal einzugrenzen.

Tabusuche-Optimierer Die Tabusuche ist ein heuristisches Optimierungsverfahren, welches an evolutionäre Algorithmen angelehnt ist. Er zeichnet sich durch seine Fähigkeit aus, nicht in lokalen Optima zu stagnieren. Dies wird durch die Möglichkeit erreicht, Teilmengen des Lösungsraums auszuschließen, die bereits berechnet wurden. Zusätzlich wird abhängig vom Verfahren eine Menge von besten Lösungen gespeichert. Nachfolgend wird der Tabusuche-Algorithmus beschrieben, wie er in [SM11] vorgestellt wird.

Das Ausschließen von Lösungen wird über eine sogenannte Tabuliste (TL) organisiert. In jedem Optimierungsschritt wird eine Menge von Nachbarlösungen generiert. Eine Nachbarlösung wird ana-

log zu EAs durch Mutation oder Rekombination erzeugt. Die Nachbarlösungen werden mit einer Evaluationsfunktion ausgewertet und absteigend sortiert und die beste Lösung ausgewählt. Danach findet der Tabutest statt, bei dem überprüft wird, ob die neue Lösung auf der Tabuliste steht. Ist dies nicht der Fall, wird sie als aktuelle Lösung akzeptiert und anschließend auf die Tabuliste gesetzt. Der Vorgang wird für alle Lösungen aus der Nachbarschaft durchgeführt. Die zuletzt als aktuell gesetzte Lösung wird für die Berechnung einer neuen Nachbarschaftsmenge verwendet.

Befindet sich eine Lösung bereits auf der Tabuliste, gibt es weiterhin die Möglichkeit, diese Lösung dennoch zu akzeptieren, wenn ein so genantes Aspirationskriterium (*aspiration level* (AV)) erfüllt wird. Das Aspirationskriterium bietet individuelle Parametrisierung, nach denen eine Lösung als *gut genug* befunden wird, um sich über den Tabustatus hinweg zusetzen.

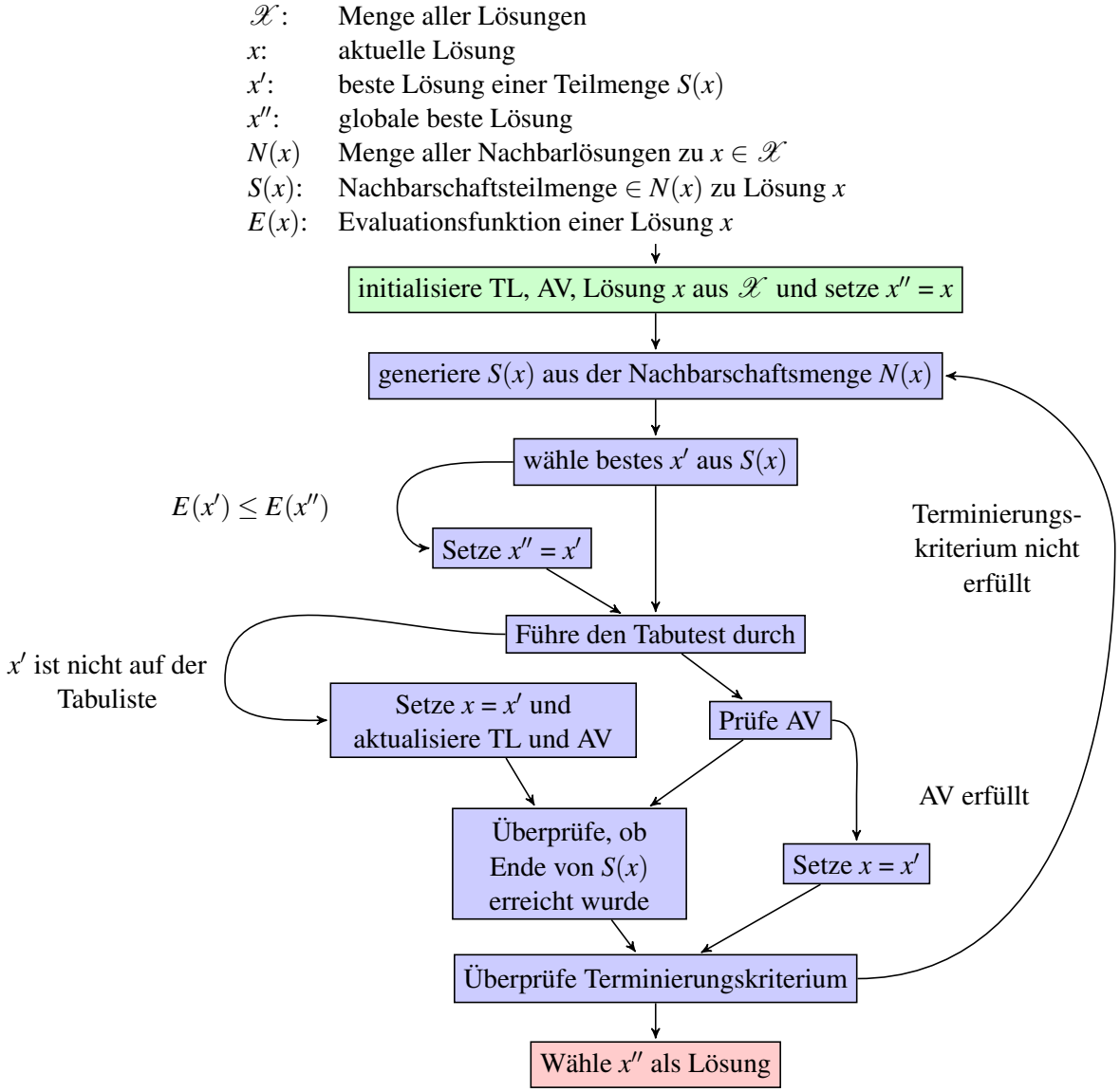


Abbildung 39: Ablauf von Tabusuche

Im *Tabusuche-Optimierer* (TS-Optimierer) von *Powder* wird ein Individuum bzw. eine Lösung der Tabusuche wie beim Simulated Annealing als ein Einsatzplan definiert. Für die Startlösung wird von

jeder Anlage ein gültiger Fahrplan ausgewählt. Die Nachbarlösung eines Individuums ist ein Einsatzplan, bei dem der Fahrplan einer einzelnen Anlage ausgetauscht wird. Dieser Austausch erfolgt ähnlich wie die Mutation des SA-Optimierers. Der Fahrplan wird durch einen Fahrplan ersetzt, bei dem in jedem Intervall die zu erzeugende prozentuale Leistung durch einen zufälligen Prozentwert ausgetauscht wurde. Die Menge der Nachbarn wird über eine Nachbarschaftsoperation ermittelt, bei der pro Anlage eine Nachbarlösung erzeugt wird. Für jede dieser Lösungen wird beim Tabutest überprüft, ob das Individuum bereits in der Tabuliste enthalten ist. Dabei wird nicht auf exakte Wertegleichheit der Fahrpläne überprüft, sondern die euklidische Distanz d berechnet. Nur wenn diese Distanz größer als ein festes ε ist, wird die Lösung auf die Tabuliste gesetzt.

Dekodierer In jedem evolutionären Optimierer werden realisierbare Einsatzpläne (Phänotypen) im Laufe der Optimierung mutiert. Diese mutierten Einsatzpläne (Genotypen) können jedoch nicht unbedingt von den Anlagen realisiert werden, da die Erzeugung der Genotypen zufallsbasiert ist und nicht die Constraints der Anlagen berücksichtigt. Somit darf die Fitness nicht anhand der Genotypen berechnet werden, da sonst Einsatzpläne bewertet würden, die von den Anlagen nicht umgesetzt werden könnten. Um evolutionäre Optimierer dennoch verwenden zu können, wird ein Dekodierer verwendet. Der Dekodierer bildet die mutierten Einsatzpläne auf realisierbare Fahrpläne ab. So können evolutionäre Optimierer verwendet und brauchbare Ergebnisse erzeugt werden.

Euklidischer Dekodierer Der euklidische Dekodierer setzt das Prinzip des Dekodierens auf einfache Weise um. Bei der Initialisierung werden zunächst alle vorhandenen Flexibilitäten übergeben. Diese Flexibilitäten werden anschließend im euklidischen Dekodierer entsprechend der einzelnen Anlagen aufgeteilt. Wird dem euklidischen Dekodierer nun ein mutierter Einsatzplan übergeben, so wird für jede Anlage aus den zuvor übermittelten Flexibilitäten der Fahrplan ermittelt, welcher dem des übergebenen Einsatzplans am nächsten kommt. Um die Nähe zueinander bestimmen zu können, wird das euklidische Distanzmaß verwendet. Somit wird im übergebenen Einsatzplan für jede Anlage der Fahrplan mit der geringsten euklidischen Distanz zurückgegeben.

Der Vorteil des euklidischen Dekodierers ist, dass die Rückgabewerte in jedem Fall von den Anlagen realisiert werden können, da die Werte direkt aus den Flexibilitäten stammen. Der Nachteil ist, dass die Menge der Rückgabewerte von der Menge der vorhandenen Flexibilitäten abhängt und somit statisch ist. Des Weiteren erhöht sich die Berechnungsdauer des Dekodierers, wenn sehr viele Flexibilitäten vorhanden sind, da bei jeder Dekodierung für jede Anlage die Distanz vom mutierten Fahrplan zu allen Fahrplänen dieser Anlage berechnet werden muss. Um diesen Nachteilen entgegenzuwirken, wurde der Supportvektor-Dekodierer entwickelt, der im nachfolgend beschrieben wird.

Supportvektor-Dekodierer Der von Dipl.-Inform. Jörg Bremer entwickelte Supportvektor-Dekodierer ermöglicht die Handhabung anlagenspezifischer Constraints bei der Veränderung von realisierbaren Fahrplänen.

Im Nachfolgenden wird auf die Art der Verwendung innerhalb des Projektes eingegangen, während eine genaue Beschreibung der Funktionsweise des Supportvektor-Dekodierer in [Bre15] nachgelesen werden kann. Zunächst musste der von ihm entwickelte Dekodierer in das Projekt integriert werden.

Als Vorbereitung wird aus einer bestehenden Menge gültiger Fahrpläne für jede Anlage (Flexibilitäten) ein Modell trainiert. Dazu werden alle Fahrpläne auf einen Wertebereich von 0 bis 1 normiert und als Trainingsdaten in den Dekodierer gegeben. Dazu wird ein Normierungsfaktor benötigt, durch den alle Werte im Fahrplan dividiert werden. Als Normierungsfaktor wird der jeweils größte Wert innerhalb eines Fahrplans verwendet. Nachdem die Modelle für jede Anlage erzeugt wurden, können beliebige normierte Fahrpläne der gegebenen Anlage auf den nächsten gültigen Fahrplan im Suchraum abgebildet werden.

Der abgebildete Fahrplan muss im Anschluss in einen nicht mehr normalisierten Fahrplan konvertiert werden. Im Bezug auf die Implementierung wurden in der Klasse `JBDecoder` die Methoden `prepareDecoder` und `mapToPossibleSchedule` geschrieben. Die erste erzeugt wie oben beschrieben für jede Anlage ein Modell und speichert dieses, während die zweite Methode einen übergebenen Fahrplan auf einen gültigen abbildet. Für die Normalisierung wurde zu den Klassen der Fahrpläne jeweils ein korrespondierender, normalisierter Fahrplan erstellt, sodass sich die beiden Fahrpläne ineinander umrechnen lassen.

6.9.4. Kombinierte Optimierung

Aus den vorigen Abschnitten 6.9.2 und 6.9.3 wurde ersichtlich, dass die EPO und die PPO getrennt voneinander als lineare Gleichungssysteme formuliert werden können. Beide Optimierungsprobleme können auch in einem gemeinsamen Gleichungssystem formalisiert und durch ein Gurobi-Programm gelöst werden. Der Vorteil dieser gemeinsamen Optimierung unter Verwendung von linearer Programmierung ist, dass garantiert die optimale Lösung des Problems gefunden wird. Nachteilig ist hingegen, wie schon an anderer Stelle erwähnt, die mit der Problemkomplexität steigende Laufzeit. Dennoch entschied sich die Projektgruppe zur Implementierung des kombinierten Optimierers, um diesen in Testszenarien, bei welchen die Laufzeit unerheblich ist, zur Überprüfung der Korrektheit der Heuristiken zu nutzen. Nachfolgend wird die Formalisierung des kombinierten Optimierungsproblems vorgestellt und die Umsetzung dessen erläutert.

Formalisierung Die folgenden für die Formalisierung notwendigen Symbole wurden in den vorigen Abschnitten bereits verwendet und werden hier der Übersichtlichkeit halber nochmals aufgeführt.

i	ein bestimmtes Marktprodukt, wobei $i \in \{0,1,\ldots,m\}$
m	Anzahl aller Block- und Stundenprodukte
$Preis_i$	Preis, für den Produkt i verkauft werden soll in $\frac{€}{MWh}$
d_i	Dauer des Produktes i in Stunden
v_i	Volumen des Produktes i pro Stunde in $\frac{€}{MWh}$
a	eine bestimmte Anlage, wobei $a \in \{0,1,\ldots,n\}$
n	Anzahl aller Anlagen im VK
j	ein bestimmter Fahrplan einer Anlage, wobei $j \in \{0,1,\ldots,s_a\}$
s_a	Anzahl der Fahrpläne der Anlage a
f_{a_j}	Fahrplan j der Anlage a
x_{a_j}	boolescher Ausdruck, der angibt, ob Fahrplan f_{a_j} gewählt wurde
$k(f_{a_j})$	Kosten für den Fahrplan f_{a_j} in $\frac{€}{MWh}$
$z(f_{a_j})$	Zusatzerlöse durch staatliche Zulagen für Fahrplan f_{a_j} in $\frac{€}{MWh}$
$l_{h_q}(f_{a_j})$	Leistung von Fahrplan f_{a_j} im Viertelstundenintervall q in Stunde h in MW, mit $h \in \{0,1,\ldots,23\}$ und $q \in \{0,1,2,3\}$
$V(h)$	Menge von Produkten, die in der Stunde h verfügbar sind

Die Zielfunktion der kombinierten Optimierung wurde bereits in Formel 13 grundlegend dargestellt. Konkret ergibt sie sich aus der Addition der Zielfunktionen der PPO und der EPO, wenn die Zielfunktion der EPO wie in der Formel 20 als Maximierungsproblem formuliert wird. Um insgesamt einen maximalen Gewinn zu erzielen, muss zum einen das Produktportfolio einen möglichst hohen Umsatz erzielen. Zum anderen sollten die Kosten für den Einsatzplan minimal sein, wobei diese zum Teil durch die staatlichen Zusatzerlöse für die Anlagen ausgeglichen werden.

$$\max\left(\sum_{i=0}^{m} Preis_i \cdot v_i \cdot d_i + \sum_{a=0}^{n}\sum_{j=0}^{s_a}\left(z\left(f_{a_j}\right) - k\left(f_{a_j}\right)\right)\cdot x_{a_j}\right)$$

Auch die Constraints ergeben sich aus den bisherigen:

- In jeder Stunde darf nur maximal so viel Energie in dem Produktportfolio eingeplant werden, wie die Anlagen erzeugen können. Dieses Constraint fügt die beiden Constraints 18 und 21 aus der PPO und der EPO zusammen: Bei der EPO ist die linke Seite der Gleichung vorgegeben (die für Stunde h im Produktportfolio eingeplante Leistung), bei der PPO die rechte (die in Stunde h erzeugbare Leistung).

$$\sum_{i \in V(h)} w_i \leq \sum_{a=0}^{n}\sum_{j=0}^{m_a} l_{h_q}(f_{a_j}) \cdot x_{a_j}, \quad \forall h \in \{0,\ldots,23\}, \quad \forall q \in \{0,\ldots,3\}$$

- Pro Anlage muss genau einer ihrer möglichen Fahrpläne gewählt werden.

$$\forall a : \sum_{j=0}^{s_a} x_{a_j} = 1$$

- Gemäß Constraint 19 muss v_i den Vorgaben des Marktes entsprechen.

$$v_i = 0\,MW \quad \vee \quad 0,1\,MW \leq v_i \leq 600\,MW$$

  Außerdem muss der Wertebereich von v_i diskret sein ($100\,kW$-Schritte).

Umsetzung In dem dargestellten Gleichungssystem sind die Variablen x_{a_j} und v_i enthalten. Die x_{a_j} sind binäre Variablen. Die v_i müssen als Integer-Variablen realisiert werden, da sie nur diskrete Werte in $100kW$-Schritten annehmen dürfen. Insgesamt ergibt sich somit ein ganzzahliges lineares Optimierungsproblem (*engl.* integer linear program). Ganzzahlige lineare Optimierungsprobleme sind NP-hart [Buc]. Dies hatte zur Folge, dass die Ausführung einer prototypischen Implementierung des kombinierten Optimierungsproblems zu inakzeptablen Laufzeiten führte. Aus diesem Grund wurde das Optimierungsproblem so vereinfacht, dass anstelle des diskreten Wertebereichs für die v_i ein kontinuierlicher Wertebereich angenommen wurde. Daraus resultiert ein gemischt-ganzzahliges (lineares) Optimierungsproblem (*engl.* mixed integer (linear) program = MIP), da es binäre und reellwertige Variablen enthält. Zwar sind gemischt-ganzzahlige Optimierungsprobleme ebenfalls NP-hart [Was05], aber trotzdem konnte eine (geringfügige) Verringerung der Komplexität des Problems erreicht werden, da die Anzahl der Integer-Variablen reduziert wurde.

Durch diese Vereinfachung des Optimierungsansatzes wurde eine Anpassung des Optimierungsergebnisses notwendig: Das Produktportfolio muss diskretisiert und der Gewinn entsprechend neu berechnet werden. Die Annahme kontinuierlicher Wertebereiche für die Produktvolumina während der Optimierung hat den Nachteil, dass der Optimierer Leistung in Produkten einplant, die aufgrund der Diskretisierungsregelung nach der Optimierung wieder korrigiert werden muss. Wäre dem Optimierer die Diskretisierungsregelung bekannt, könnte er diese Leistung gegebenenfalls in anderen Produkten gewinnbringender einplanen. Somit findet auch der kombinierte lineare Optimierer nicht garantiert die optimale Lösung.

6.9.5. Implementierung

Die Implementierung der Optimierung muss die in Kapitel 6.9.1 erläuterte Abhängigkeit der Optimierungsprobleme berücksichtigen. Entweder wird zuerst ein Einsatzplan erstellt und anschließend ein dazu passendes Produktportfolio erzeugt, oder zunächst ein Produktportfolio generiert, welches durch einen darauf folgend zu erstellenden Fahrplan abgedeckt werden muss. Um die Erweiterbarkeit zu gewährleisten und beide Überlegungen umsetzen zu können, wurden abstrakte Optimiererklassen entwickelt, welche die Realisierung beider Varianten ermöglichen. In Abbildung 40 ist das Klassendiagramm der abstrakten Optimiererklassen zu sehen.

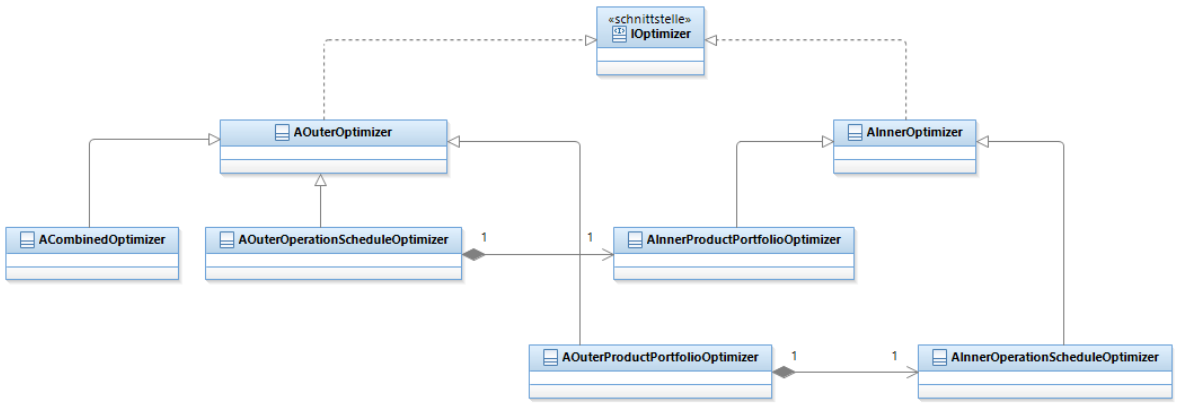


Abbildung 40: Optimierer Klassendiagramm

Alle Optimierer implementieren das Interface `IOptimizer` und erweitern entweder `AOuterOptimizer` oder `AInnerOptimizer`.

Dabei sind innere Optimierer Bestandteile der äußeren Optimierer, welche in einer 1:1 Beziehung zueinander stehen. Die abstrakte Trennung zwischen innerem und äußerem Optimierer ermöglicht eine feinere Konkretisierung in der nächsten Abstraktionsebene. Hier wird die Umsetzung beider Optimierungsvarianten ermöglicht. Der im Klassendiagramm abgebildete `AOuterOperationScheduleOptimizer` wird verwendet, wenn zunächst ein Einsatzplan erstellt wird, zu dem anschließend vom `AInnerProductPortfolioOptimizer` ein passendes Produktportfolio erzeugt wird. Bei der Umsetzung der zweiten Optimierungsvariante ist der Produktportfoliooptimierer der äußere und der Einsatzplanoptimierer der innere Optimierer. In diesem Fall ist der `AInnerOperationScheduleOptimizer` Bestandteil des `AOuterProductPortfolioOptimizer`. Dieser abstrakte Lösungsansatz ermöglicht einen flexiblen Austausch der inneren und äußeren Optimierer. In den beiden Kapiteln 6.9.2 und 6.9.3 wird die Zuordnung der Optimierungen erläutert. Aufgrund der Problemkomplexität der EPO wird diese mittels einer Heuristik gelöst und als äußere Optimierung verwendet. Dem entsprechend agiert die PPO als innere Optimierung. Diese Zuordnung kann in einer Weiterentwicklung von *Powder* bei Bedarf geändert werden.

Eine gesonderte Rolle nimmt in dieser Architektur der `ACombinedOptimizer` ein. Der `ACombinedOptimizer` ist ebenfalls ein `AOuterOptimizer`, jedoch ohne korrespondierenden inneren Optimierer. Hier werden beide Optimierungsprobleme nicht in äußere und innere Optimierung getrennt, sondern als ganzes betrachtet. In *Powder* wird der `ACombinedOptimizer` durch den linearen Optimierer realisiert, welcher in Abschnitt 6.9.4 genauer erläutert wurde.

Ein- und Ausgabedatenstruktur Die Datenstrukturen, die das Optimierungsmodul bei der Kommunikation mit den übrigen Modulen nutzt, sind die Anlagenflexibilitäten (s. Abschnitt 6.8.1), die Marktprognose (s. Abschnitt 6.7) und das `OptimizationResult`. Dabei stellen die Flexibilitäten und die Marktprognose Eingabedaten dar. Als Ausgabedatenstruktur wurde das `OptimizationResult` entwickelt. Dieses beinhaltet das `ProductPortfolio` (*dt.* Produktportfolio) und den `OperationSchedule` (*dt.* Einsatzplan). Eine detailliertere Beschreibung des `OptimizationResult` kann in Anhang E.3 nachgelesen werden.

Nutzung des Builder Patterns Um die konkrete Optimiererumsetzung variabel und erweiterbar zu halten, wurde das Builder Pattern umgesetzt. Die Klasse `OptimizerBuilder` bietet Methoden zur Erstellung beliebiger Optimierer an, welche über eine `OptimizerDescription` definiert sind. Diese `OptimizerDescription` enthält notwendige Parameter bzw. Komponenten zur Erstellung der Optimierer, wie beispielsweise der gewählte innere Optimierer (wiederum eine `OptimizerDescription`). Eine `OptimizerDescription` kann automatisch aus der Klasse des Optimierers (gekennzeichnet durch entsprechend typisierende Interfaces) generiert werden, da die notwendigen Parameter und Komponenten über eine Annotation gekennzeichnet werden. Falls ein neues Optimierungsverfahren implementiert werden soll, so kann dieses direkt über das bestehende Programm genutzt werden, sofern die Implementierung sich an das kennzeichnende Schema hält. Diese Vorlage wird dann zur Laufzeit mit den gewählten Parametern und Komponenten gefüllt (was prinzipiell auch durch den Nutzer geschehen könnte) und dient der Klasse `OptimizerBuilder` als Parameter- und Komponentensammlung, welche die einzelnen Komponenten für einen bestimmten Optimierer darstellt. Dieser wird dann unabhängig von der spezifischen Implementierung aus der Sammlung konstruiert. Da für die Erfüllung der qualitativen Anforderung (Q-01) einzig der beste Optimierer genutzt werden muss, wurde die variable Optimiererauswahl für den Betrieb eingeschränkt auf die durch die Evaluation als beste angenommene Optimiererkombination.

6.9.6. Metaoptimierung

Heuristische Optimierungsverfahren sind meistens von Parametern abhängig, mit denen der Optimierungsprozess in bestimmte Richtungen gesteuert werden kann. Bei evolutionären Algorithmen, wie z. B. dem *Simulated Annealing*, sind die Mutationsstärke σ und, speziell beim SA, die Anfangsstemperatur T_0 solche Parameter. Ist die Mutationstärke zu gering, wird im Suchraum nur lokal nach der besten Lösung gesucht und abhängig vom Optimierungsverfahren kann das lokale Optimum nicht verlassen werden. Bei hoher Mutationsstärke hingegen wird der Suchraum mit großen Schritten abgesucht. Diese Schritte sind allerdings ungenauer, sodass zwar gute Lösungen gefunden werden können, diese aber nicht unbedingt auch die besten Lösungen sein müssen. Die Wahl des optimalen Wertes für σ ist folglich entscheidend für die Lösungsqualität.

Generell gibt es dafür zwei Möglichkeiten. Mittels Parametersteuerung wird ein Parameter, wie die Mutationsstärke, während der Laufzeit angepasst und mit Parameteroptimierung wird der beste Wert eines Parameters experimentell bestimmt. Bei letzterem wird die Optimierung mehrfach mit verschiedenen Werten des Parameters durchgeführt und der Wert ausgewählt, der zu den besten Ergebnissen geführt hat.

Metaoptimierung des SA-Optimierers Der SA-Optimierer besitzt, wie oben erwähnt, die beiden Parameter Mutationsstärke und Anfangstemperatur. Diese werden jedoch, wie im Abschnitt 6.9.3 beschrieben, über die Parametersteuerung festgelegt, sodass keine Parameteroptimierung notwendig ist.

Metaoptimierung des TS-Optimierers Der TS-Optimierer in *Powder* hat zwei Parameter, zum einen die Größe der Tabuliste und zum anderen der Grenzwert ε, der angibt, ob eine Lösung bereits

in der Tabuliste enthalten ist. Die Metaoptimierung dieser Parameter wurde mit Hilfe der Infrastruktur des Evaluationsmoduls (s. Abschnitt 6.10) durchgeführt. Im Vorfeld wurden für jeden Parameter Werte festgelegt, die bei der Metaoptimierung betrachtet werden sollen. Dabei wurde darauf geachtet, dass sich die Werte ausreichend unterscheiden, um einen Einfluss auf die Lösungsqualität zu bewirken. Der Wert ε ist in Prozent bzw. der entsprechenden Dezimalrepräsentation anzugeben. Hier wurde das Intervall von 5% bis 25% gleichmäßig aufgeteilt. Um signifikante Änderungen zu bewirken, wurde die Größe der Tabuliste in Schritten mit dem Faktor zehn betrachtet, also die Werte 10, 100, 1.000, 10.000 und 100.000 verwendet. Das Ergebnis mehrere Durchläufe ist in Abbildung 41 zusammengefasst.

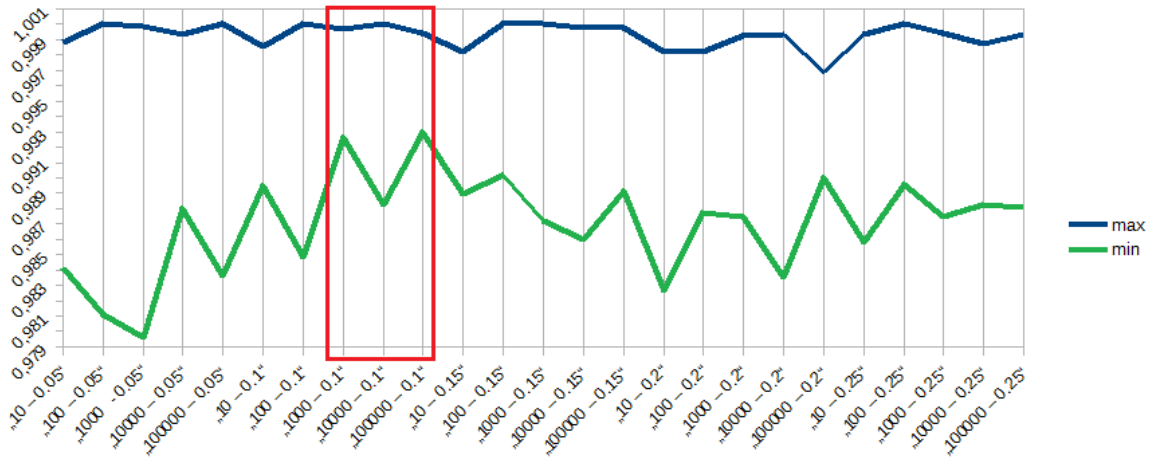


Abbildung 41: Ergebnisse der Metaoptimierung des TS-Optimierers. Auf der y-Achse befinden sich normalisierte Fitnesswerte, wobei der beste Wert als Referenz dient. Auf der x-Achse befinden sich jeweils Kombinationen der eingestellten Parameter.

Während die Maximalwerte in einem sehr geringen Bereich schwanken, sind bei den Minimalwerten deutlichere Unterschiede zu erkennen, weshalb diese für die Suche nach den optimalen Parameterwerten mit einbezogen werden. Die besten Ergebnisse sind demnach im markierten Bereich zu finden. Dies entspricht einem ε-Wert von 10% und Tabulistengröße von 1.000 bzw. 100.000. Da weiterhin die Tendenz zu erkennen ist, dass größere Tabulisten bessere Ergebnisse liefern, wurde der Wert 100.000 als Größe für die Tabuliste festgelegt.

6.9.7. Angebotsberechnung

Für den Verkauf an der EPEX SPOT Strombörse muss zu jedem Paket auch der angebotene Preis übergeben werden. Der Preis, für den ein Paket verkauft wird, ermittelt EPEX SPOT abhängig von Angebot und Nachfrage, wie im Seminarband in Kapitel 4 (s. Abschnitt G) beschrieben wird. Falls der Angebotspreis unter dem so ermittelten Verkaufspreis liegt, wird das Paket verkauft. Der erwartete Verkaufspreis wird von der Marktprognose berechnet. Je kleiner der Angebotspreis ist, desto höher ist die Verkaufswahrscheinlichkeit. Der Angebotspreis sollte also geringer als dieser Verkaufspreis, aber größer als die Kosten der Anlagen sein, damit ein Gewinn erzielt werden kann.

Für die Berechnung des Angebotspreises wird eine Normalverteilung angenommen, da keine konkreten Daten, die auf eine spezielle Verteilung schließen lassen könnten, vorliegen. Der Erwartungswert wird von der Marktprognose berechnet. Die Breite der Normalverteilung wird anhand der Standardabweichung der Marktprognose zu den tatsächlichen Marktdaten eingestellt. Dazu wird die Standardabweichung über einen größeren Zeitraum gemittelt. Die Auswahl des Angebotspreises wird anhand der Abbildung 42 verdeutlicht. Dazu sei angemerkt, dass die Werte der X-Achse (in €/MWh) der Übersicht halber relativ zum Erwartungswert angegeben sind.

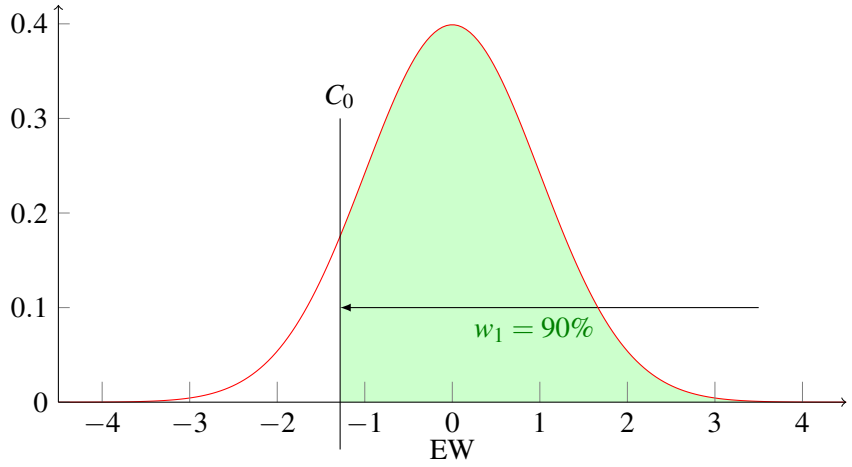


Abbildung 42: Angebotsberechnung: Normalverteilung um den Erwartungswert (EW)

Zuerst wird eine Wahrscheinlichkeit w_1 angegeben, mit der der vorläufige Angebotspreis C_0 unter dem tatsächlichen Verkaufspreis liegt (d. h. in diesem Beispiel liegt 90 % der Fläche der Gaußkurve rechts von C_0). Falls die Anlagenkosten K den vorläufigen Angebotspreis C_0 um einen Mindestgewinn g unterschreiten, wird C_0 als tatsächlicher Angebotspreis C ausgewählt. Andernfalls wird C aus der Summe der Kosten und des Mindestgewinns bestimmt ((23)).

$$C = \max\{K + g, C_0\} \tag{23}$$

Als nächstes wird auf die Implementierung der Angebotsbrechung eingegangen. Dazu werden nach Ermittlung des Produktportfolios alle darin erhaltenen Marktprodukte zusätzlich mit einem Angebotspreis versehen. Dieser Vorgang wird in der Klasse `OfferPriceCalculator` implementiert und erwartet in der Methode `calculateOfferPrice` ein `OptimizationResult`, sowie eine Verkaufswahrscheinlichkeit (w_1). Initial wird die Standardabweichung für jedes Marktprodukt aus einer Datei ausgelesen, die vorher durch das Marktmodul berechnet wurde. Da jedes Marktprodukt einzeln betrachtet werden muss, werden in einer Schleife alle Marktprodukte aus dem Produktportfolio durchgegangen. Mit Hilfe der Standardabweichung und dem Erwartungswert kann eine Normalverteilung erzeugt werden. Zur Erzeugung der Normalverteilung wird die Klasse `NormalDistribution` aus dem Paket *org.apache.commons.math3* zur Hilfe genommen. Anschließend kann über die Methode `inverseCumulativeProbability` der vorläufige Angebotspreis (C_0) unter Berücksichtigung der Verkaufswahrscheinlichkeit für das Marktprodukt berechnet werden. Zusätzlich werden die Kosten

auf die einzelnen Pakete verteilt, sodass zu dem C_0 eine weitere Grenze K entsteht. Der Angebotspreis wird entsprechend der obigen Formel (23) gesetzt und in dem Marktprodukt gespeichert.

6.9.8. Abweichungen zur Anforderungsdefinition

Die Mehrheit der in der Analysephase erhobenen Anforderungen an die EPO und die PPO wurde abweichend zu ihrer ursprünglichen Definition realisiert. Dafür gibt es drei Gründe. Zum einen war zum Zeitpunkt der Anforderungserheben noch unklar, in welcher Reihenfolge die beiden Optimierungsprobleme gelöst werden würden. Der Einsatzplanoptimierer wurde als äußerer, heuristischer Optimierer realisiert. Er solcher erzeugt er in jedem Optimierungsschritt zufällige Einsatzpläne und muss daher entgegen der Anforderung EPO-PUC2 kein initiales Referenzanlagenportfolio erstellen. Aus demselben Grund wurde auch Anforderung EPO-PUC4 nicht umgesetzt. Der Einsatzplanoptimierer mutiert zwar den in der vorigen Iteration erstellten Einsatzplan, allerdings wird dieser nicht wie in der Anforderung gefordert entsprechend der Lastkurve des Produktportfolios optimiert, sondern nur indirekt über die Fitness des mutierten Einsatzplans, die auch den Umsatz durch das Produktportfolio berücksichtigt. Eine weitere abweichend umgesetzte Anforderung ist (EPO-PUC3): Der Einsatzplanoptimierer erhält keine Lastkurve vom Produktportfoliooptimierer, sondern das gesamte Produktportfolio. Die Lastkurve ist nicht erforderlich, da aus oben genannten Gründen aus dem Ergebnis des Produktportfoliooptimierers nur die Fitness des erstellten Produktportfolios verwendet wird. Die Nutzung des Produktportfoliooptimierers als innereren Optimierers bewirkt, dass dieser aus dem Einsatzplan nur die Lastkurve der mindestens zu erzeugenden Energie aber nicht die Einsatzplankosten (Anforderungen EPO-PUC5 und PPO-PUC1) benötigt. Deshalb ist auch keine initiale Referenzleistungskurve notwendig; Der Produktportfoliooptimierer erhält in jedem Optimierungsschritt vom äußeren Einsatzplanoptimierer die zu erreichende Lastkurve PPO-PUC7.

Weiterhin wurden Anforderungen nicht umgesetzt, da diese nicht mehr relevant waren. Anforderung PPO-PUC6 fordert, dass der Produktportfoliooptimierer das Produktportfolio an den Direktvermarkter sendet. Stattdessen erhält der Einsatzplanoptimierer das Produktportfolio und übergibt es an das VPP-Modul. Auch dieses sendet das Produktportfolio nicht an den Direktvermarkter, da *Powder* die Tätigkeit des Direktvermarkters (Zusammenstellung des Produktportfolios) ersetzen soll. Die Einreichung des Produktportfolios zur Auktion obliegt dem VK-Betreiber (s. Abschnitt 1.1). Auch Anforderung PPO-PUC3, das Festlegen von Verkaufspreisen für die Produkte, wurde im Optimierungsmodul nicht umgesetzt (Begründung s. Abschnitt 6.9.2). Während der Optimierung werden die prognostizierten Marktpreise verwendet. Erst nach der Optimierung wird berechnet, bei welchen Verkaufspreisen das Produktportfolio mit welcher Wahrscheinlichkeit verkauft wird.

Ein letzter Grund für nicht umgesetzte Anforderungen war, dass durch Implementierungsentscheidungen aktiv formulierte Anfragen durch passive Parameterübergabe ersetzt wurden. Dies ist bei den Anforderungen EPO-PUC1 und PPO-PUC2 der Fall.

6.10. Evaluationsmodul

Das Ziel der Evaluation war, eine Aussage darüber treffen zu können, welcher der entwickelten Optimierer bei welcher Konstellation von Parameterwerten (Szenario), die eine konkrete Konfiguration des VK repräsentiert, der am meisten geeignete ist oder ob einer der Optimierer für alle Fälle der beste ist. Da für die Gesamtoptimierung jeweils ein innerer Optimierer und ein äußerer Optimierer benötigt werden, wurden zunächst die inneren Optimierer gegeneinander evaluiert und anschließend die äußeren mit jeweils dem besten der inneren Optimierer ausgeführt. Der Evaluation der Optimierer ging die Festlegung optimaler Parameterwerte der heuristischen Optimierer voran (s. Abschnitt 6.9.6).

In diesem Abschnitt wird zunächst eine generelle Einführung in den Entwurf von Evaluationsstrategien gegeben, bevor die Qualitätskriterien für die Optimierer, die untersuchten Parameter und die konkrete Ausgestaltung des Evaluationsablaufs vorgestellt werden. Anschließend werden die Ergebnisse der Evaluation aufgezeigt und die Implementierung beschrieben.

6.10.1. Design of Experiments

Der Begriff *Design of Experiments* bezeichnet den Entwurf der Evaluationsstrategie für ein System. Dabei wird das Ziel verfolgt, die Systemeigenschaften mit möglichst wenig Durchläufen zu untersuchen. Nach [NISb] wird beim Design of Experiments das zu evaluierende System als Black-Box aufgefasst, das Eingabeparameter (*Faktoren*) und Ausgabeparameter (*Antworten*) besitzt. Zur Evaluation des Systems wird jedem Faktor ein Wert (*Faktorstufe*) zugewiesen und das Systemverhalten mit Hilfe der Antworten analysiert. Ein solcher Durchlauf wird als *Experiment* bezeichnet. Um allgemeine Aussagen über das Systemverhalten treffen zu können, ist eine Vielzahl von Experimenten mit jeweils anderen Faktorstufen notwendig. Da das Durchführen von Experimenten für jede mögliche Kombination von Faktorfstufen aus zeitlichen Gründen oftmals nicht realisierbar ist, muss eine Auswahl von Experimenten (*Design*) vorgenommen werden. In der Literatur existieren verschiedene Ansätze, um eine geeignete Auswahl finden zu können. Welcher der Ansätze verwendet werden sollte, ist davon abhängig, welches Ziel mit der Evaluation verfolgt wird. Im Folgenden werden drei bekannte Designs vorgestellt.

Vollfaktorielles Design Beim vollfaktoriellen Design (engl. full factorial) wird für jede mögliche Kombination der Wertebelegung der Faktoren ein Experiment durchgeführt [Frab, NISa]. Es hat den Vorteil, dass Wechselwirkungen zwischen den Faktoren sehr gut analysierbar sind. Nachteilig ist allerdings die sehr große Anzahl an Experimenten, die benötigt wird, um alle Kombinationen abzudecken: Für n Faktorstufen und k Faktoren ergeben sich n^k Experimente. Beispielsweise müssten bei einem System mit sieben Faktoren und nur zwei Faktorstufen bereits $2^7 = 128$ Experimente durchgeführt werden.

Teilfaktorielles Design Beim teilfaktoriellen Design (engl. fractional factorial) wird die Anzahl an Experimenten gegenüber dem vollfaktoriellen Design reduziert, indem für einen Teil der Faktoren deren Faktorstufen abhängig von den Faktorstufen der übrigen Faktoren formuliert werden [Fraa]. Ein bekannter Algorithmus zum Erstellen und Auswerten teilfaktorieller Designs ist der Yates-

Algorithmus. Dieser ermöglicht es, Schätzwerte für die Effekte der einzelnen Faktoren zu berechnen [NISc]. Der Yates-Algorithmus wird oft bei Experimenten mit nur zwei Faktorstufen verwendet, ist generell aber auch bei mehr Faktorstufen anwendbar [Spl02].

Latin Hypercube Latin Hypercube ist eine Sampling-Methode, bei der aus dem Raum aller möglichen Kombinationen der Faktorstufen zufällig so viele ausgewählt werden, wie Experimente durchgeführt werden sollen. Um eine gute Suchraumabdeckung zu erreichen, geschieht die Auswahl nicht rein zufällig, sondern unterliegt gewissen Regeln. Diese werden folgend an einem Beispiel mit zwei Faktoren veranschaulicht.

Zunächst werden die Wertebereiche der Faktoren in i Intervalle unterteilt. Die Anzahl der Intervalle ist für jeden Faktor gleich und entspricht der Anzahl an durchzuführenden Experimenten. Auf diese Weise bildet sich bei zwei Faktoren ein Quadrat mit $i \times i$ Zellen. Anschließend werden i Zellen zufällig ausgewählt, jedoch so, dass aus jeder Zeile und jeder Spalte genau eine Zelle ausgewählt wird, damit der Datenraum gut abgedeckt wird [CD00]. Dieses Prinzip lässt sich auf den Fall übertragen, wenn mehr als zwei Faktoren vorliegen: Diese bilden dann einen Hyperwürfel.

Der Vorteil von Latin Hypercube ist, dass die Anzahl an Experimenten nicht, wie beim vollfaktoriellen Design, von der Anzahl an Faktoren abhängt, sondern beliebig gewählt werden kann. Allerdings lassen sich die Zusammenhänge zwischen den Faktoren nicht analysieren.

Auswahl des Designs Im Rahmen der Projektgruppe war es weniger das Ziel der Evaluation, quantitative Aussagen über die Stärke der Effekte der einzelnen Faktoren auf die Ausgaben zu machen, als vielmehr qualitativ zu beurteilen, ob sich das Verhalten der Optimierer bei verschiedenen Kombinationen von Faktorstufen sichtbar ändert. Aus diesem Grund wurde Latin Hypercube als Design verwendet. Um die Auswirkungen einzelner Parameter, die bei der Auswertung der mit Latin Hypercube entworfenen Experimente relevant erschienen, genauer untersuchen zu können, wurde ebenfalls eine Implementierung des vollfaktoriellen Designs erstellt (6.10.6).

6.10.2. Qualitätskriterien

Die Qualität eines Optimierers wird durch verschiedene Qualitätskriterien definiert. Für jeden der Optimierer wurde anhand dieser Qualitätskriterien untersucht, ob unterschiedliche Konstellationen der Parameterwerte Auswirkungen auf dessen Verhalten haben. Die betrachteten Qualitätskriterien sind die folgenden:

absolute Fitness Die absolute Fitness gibt für die inneren Optimierer den Umsatz an, der durch das Produktportfolio am Markt erzielt werden kann. Bei den äußeren und dem kombinierten Optimierer gibt die absolute Fitness den Gewinn an, der nach Einbezug der Kosten und Zuschläge insgesamt erreicht werden kann.

Um die Auswirkungen von Zufallseffekten zu minimieren, wurde jedes Experiment mehrmals wiederholt. Daher berechnet dieses Kriterium über alle Wiederholungen den Durchschnitt der Fitness.

Lösungsgüte Die Lösungsgüte eines Optimierers ist eine relative Fitness, die sich wie folgt berechnet:

- für äußere Optimierer:
  - falls die optimale Fitness vom kombinierten linearen Optimierer berechnet werden konnte: $\frac{\text{absolute Fitness}}{\text{optimale Fitness}}$
  - sonst: $\frac{\text{absolute Fitness}}{\text{absolute Fitness des besten äußeren Optimierers}}$
- für innere Optimierer: $\frac{\text{absolute Fitness}}{\text{absolute Fitness des besten inneren Optimierers}}$

Standardabweichung der Fitness Die Standardabweichung der Fitness wird ebenfalls über alle Wiederholungen eines Experiments berechnet. Sie ist ein Maß für die Zuverlässigkeit des Optimierers: je geringer die Standardabweichung, desto höher die Zuverlässigkeit.

Die vorgestellten Qualitätskriterien gelten sowohl für die inneren als auch für die äußeren Optimierer. Weiterhin wurden für innere und äußere Optimierer jeweils spezifische Kriterien betrachtet:

Anzahl Fitnessfunktionsaufrufe für äußere Optimierer. Gezählt wird, wie oft ein Optimierer seine Fitnessfunktion während eines Experiments aufruft. Dies Qualitätskriterium wird als Maß dafür gesehen, wie ressourcenschonend ein Optimierer ist, da die Berechnung der Fitness sehr aufwendig ist: Jedes Mal folgt ein Aufruf des inneren Optimierers. Für die inneren Optimierer ist dies Qualitätskriterium unerheblich, da diese ohne Fitnessfunktion arbeiten.

Laufzeit für innere Optimierer. Spielt bei den äußeren Optimierern keine Rolle, da diese einen spätesten Endzeitpunkt übergeben bekommen, bis zu welchem sie die Optimierung beendet haben müssen. Diese Zeit nutzen die äußeren Optimierer vollständig aus.

6.10.3. Untersuchte Faktoren

Die folgenden Faktoren wurden für die Evaluation als relevant erachtet:

Anzahl der Anlagen im VK entspricht der Summe der Anzahl von PV- und BHKW-Anlagen.

Gesamtleistung des VK entspricht der Summe der Maximalleistungen der einzelnen Anlagen.

Anteil der Leistung der BHKW-Anlagen an der Gesamtleistung des VK legt fest, dass beispielsweise 20% der Leistung des VK von den BHKW-Anlagen erbracht werden muss. Da in *Powder* zur Zeit nur BHKW- und PV-Anlagen verwendet werden, müssen die restlichen 80% der Leistung von den PV-Anlagen erbracht werden.

Anzahl an Fahrplänen pro BHKW-Anlage legt fest, wie viele Fahrpläne für eine BHKW-Anlage erzeugt werden sollen. Da PV-Anlagen nur genau einen möglichen Fahrplan besitzen, ist für diese kein weiterer Faktor notwendig.

Ein weiterer Einflussfaktor auf die Qualität der äußeren Optimierer ist die Laufzeit, die diesen zur Verfügung gestellt wird. Falls sie über mehr Zeit verfügen, können sie den Suchraum großflächiger durchsuchen und somit wahrscheinlicher auf die optimale Lösung stoßen. Die Laufzeit wurde allerdings nicht beim Latin Hyercube Sampling berücksichtigt. Stattdessen wurde eines der mit Latin Hypercube generierten Designs für verschiedene Laufzeiten durchgeführt (s. Schritt 2b der Evaluation der äußeren Optimierer).

Das Datum des Tages, für den die Optimierung durchgeführt wird, kann ebenfalls variiert werden. Es wurde jedoch nicht als Faktor in das Latin Hypercube Sampling aufgenommen, da es keinen Einfluss auf das Verhalten der Optimierer hat. Es beeinflusst lediglich die prognostizierten Marktpreise und die Flexibilitäten der Anlagen, da diese wetter- und temperaturabhängig sind. Durch die Variation des Datums kann somit beispielsweise untersucht werden, zu welcher Jahreszeit ein VK-Betreiber mit welchem Anlagentyp den größten Gewinn erzielen kann. Für die Evaluation der Optimierer wurde willkürlich der 15.01.2012 ausgewählt.

6.10.4. Ablauf der Evaluation

Die Evaluation der Optimierer wurde in mehreren Schritten durchgeführt. Zunächst wurden die inneren Optimierer evaluiert, damit bei der Evaluation der äußeren Optimierer der beste innere eingesetzt werden konnte. Für die Evaluation der heuristischen äußeren Optimierer wurden mehrere Schritte benötigt, da ihr Verhalten komplexer als das der inneren Optimierer ist:

1. Auswahl des am besten geeigneten Dekodierers
2. Allgemeiner Vergleich der Optimierer
   a) Auswertung der Standardabweichung der Fitness
   b) Auswertung der Fitness in Abhängigkeit von der Laufzeit
   c) Auswertung der Lösungsgüte
   d) Auswertung der Anzahl an Fitnessfunktionsaufrufen
3. Untersuchung ausgewählter Faktoren
   a) Untersuchung der Fahrplananzahl
   b) Untersuchung der VK-Leistung und Anlagenanzahl

6.10.5. Auswertung

In diesem Abschnitt werden die Ergebnisse der Evaluationsschritte präsentiert. Eine vollständige Übersicht über die Wertebelegungen der Faktoren in den einzelnen Experimenten kann in Anhang A eingesehen werden.

Evaluation der inneren Optimierer Für die Evaluation der inneren Optimierer wurde ein Latin Hypercube Design mit zehn Experimenten verwendet. Die Wertebereiche der Faktoren wurden wie folgt gewählt:

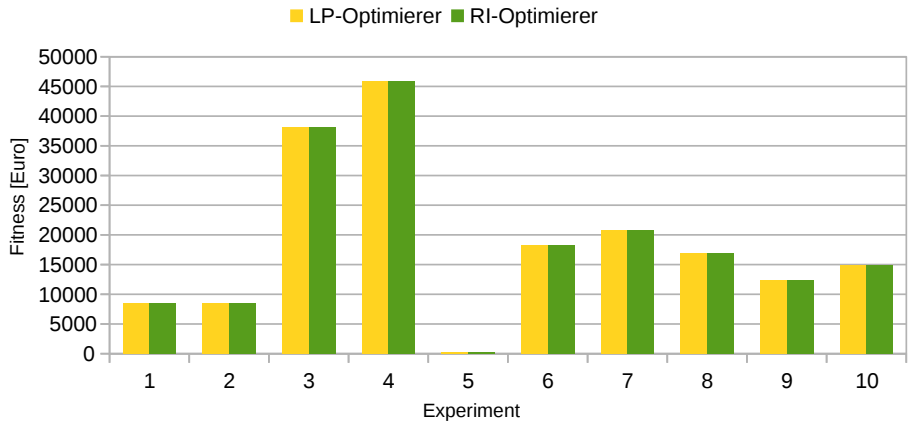


Abbildung 43: Fitness der inneren Optimierer

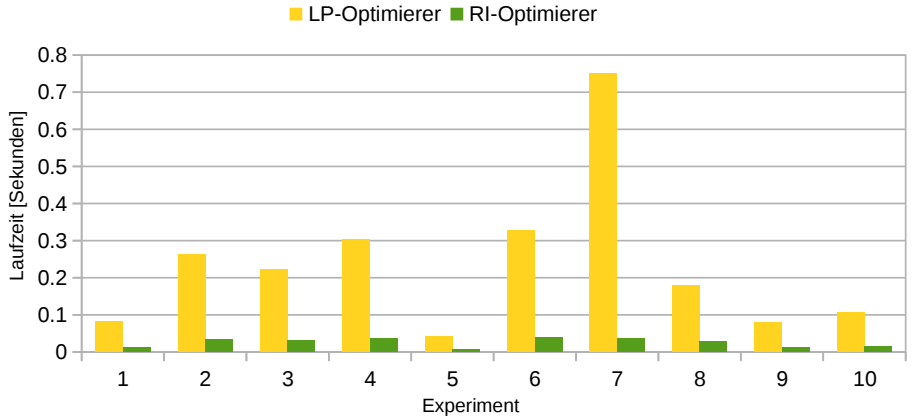


Abbildung 44: Laufzeit der inneren Optimierer

Anzahl der Anlagen im VK	$\in \{1;\ldots;5000\}$
Gesamtleistung des VK [kWh]	$\in \{1;\ldots;5000\}$
Anteil der BHKW-Leistung an der Gesamtleistung des VK	$\in \{0,0;\ldots;1,0\}$
Anzahl an Fahrplänen pro BHKW-Anlage	$\in \{1;\ldots;300\}$

Die Ergebnisse der Experimente sind in den Abbildungen 43 und 44 dargestellt. Sie zeigen, dass der RI-Optimierer und der LP-Optimierer bei jedem Experiment dieselbe Fitness liefern, wobei der RI-Optimierer jedesmal eine deutlich kürzere Laufzeit aufweist. Da die Laufzeitunterschiede mit weniger als einer Sekunde im Verhältnis zur Dauer der Gesamtoptimierung unerheblich sind, eignen sich beide inneren Optimierer für den Realbetrieb von *Powder*. Dennoch wurde der RI-Optimierer aufgrund seiner geringeren Laufzeit in den weiteren Evaluationschritten eingesetzt.

Die Gleichheit der Fitnesswerte erklärt sich dadurch, dass lineare Optimierungsverfahren immer die optimale Lösung liefern und auch der RI-Optimierer deterministisch arbeitet, um die beste Lösung zu finden. Der RI-Optimierer kommt mit einer kürzeren Laufzeit aus, da er spezifisch auf das vorliegende Optimierungsproblem zugeschnitten ist, während der LP-Optimierer Gleichungssysteme lösen muss.

Evaluation der äußeren Optimierer In diesem Teil wird evaluiert, welcher äußere Optimierer mit welchem Dekodierer die am besten geeignete Konfiguration darstellt. Zusätzlich wird der Einfluss ausgewählter Faktoren auf diese Konfiguration betrachtet.

1. **Auswahl des am besten geeigneten Dekodierers**
   Für die äußeren Optimierer wurden zwei verschiedene Dekodierer implementiert: der euklidische Dekodierer und der Supportvektor-Dekodierer (s. Abschnitt 6.9.3). Für die Evaluation der Dekodierer wurde ein Latin Hypercube Design mit fünf Experimenten erstellt, welches für jeden der Optimierer mit beiden Dekodierern durchgeführt wurde. Um Zufallseffekte der Dekodierer auszugleichen, wurde das Design jeweils zehnmal wiederholt. Die Werte der Faktoren wurden folgendermaßen gesetzt:

Anzahl der Anlagen im VK	$\in \{1;\ldots;100\}$
Gesamtleistung des VK [kWh]	$\in \{100;\ldots;3000\}$
Anteil der BHKW-Leistung an der Gesamtleistung des VK	$1,0$
Anzahl an Fahrplänen pro BHKW-Anlage	$\in \{2;\ldots;500\}$

   Es wurden keine PV-Anlagen eingesetzt, da diese nur einen möglichen Fahrplan besitzen. Aus diesem Grund wird ihr Fahrplan von den Optimierern nicht mutiert, sodass sich beide Optimierer bezüglich PV-Anlagen gleich verhalten. PV-Anlagen haben damit auf die Performanz des Dekodierers keinen Einfluss und wurden deshalb nicht betrachtet.

   Für jede Wiederholung eines Experimentes wurde den Optimierern eine Laufzeit von zehn Sekunden zur Verfügung gestellt. Längere Optimierungszeiträume sind für diesen Evaluationsschritt nicht notwendig, da sie zwar wahrscheinlich zu besseren Fitnesswerten der Optimierer führen würden, diese aber lediglich durch die Funktionsweise der Optimierer bedingt sind und somit nicht mehr Aussagen über die Dekodierer zulassen, als es kürzere Zeiträume ebenfalls ermöglichen.

   In allen Experimenten, außer einem, erzielten die Optimierer mit dem Supportvektor-Dekodierer eine schlechtere Fitness als mit dem euklidischen Dekodierer. In Abbildung 45 ist die Änderung der Fitness (in Prozent) angegeben, die sich für einen Optimierer ergibt, wenn der Supportvektor-Dekodierer statt des euklidischen verwendet wird. Sie wurde in Relation zur besseren Fitness des jeweiligen Optimierers in dem Experiment berechnet:

$$\frac{\text{Fitness mit Supportvektor-Dekodierer} \;-\; \text{Fitness mit euklidischem Dekodierer}}{\text{Maximum beider Fitnesswerte}} \cdot 100$$

   Die Verschlechterung der Fitness bewegt sich mit bis zu 2% aber in einem relativ geringen Bereich. In einem Experiment verringerte sich die Fitness jedoch um fast 12%.

   Ein weiterer Nachteil des Supportvektor-Dekodierers ist, dass der SA-Optimierer mit ihm bei den Experimenten, in denen die BHKW-Anlagen über mehr als 360 Fahrpläne verfügten, NaN-Werte als Fitness erzeugte. Aus diesen beiden Gründen wurde der euklidische Dekodierer bei den weiteren Evaluationsschritten eingesetzt und als Dekodierer für den Betrieb gewählt.

2. **Allgemeiner Vergleich der Optimierer**

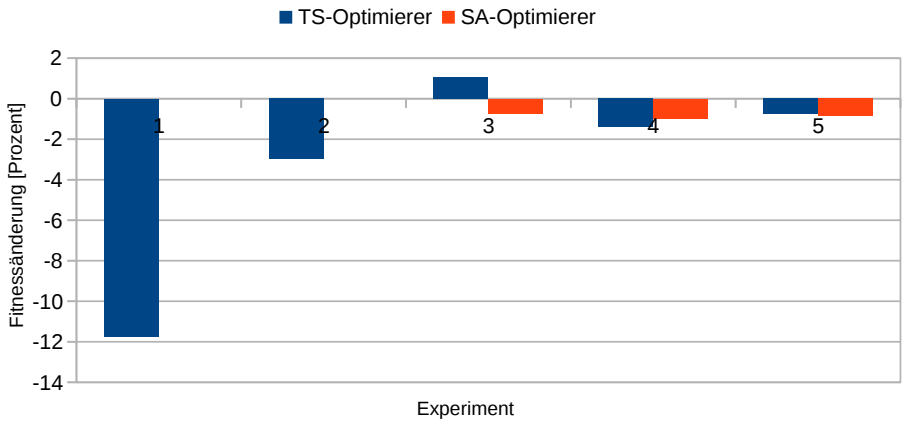


Abbildung 45: Änderung der Fitness bei Verwendung des Supportvektor-Dekodierers statt des euklidischen Dekodierers (fehlende Werte des SA-Optimierers sind NaN)

In diesem Teil werden die beiden Optimierer gegeneinander evaluiert. Die Kriterien umfassen die Standardabweichung der Fitness, die Fitness in Abhängigkeit von der Laufzeit, die allgemeine Lösungsgüte sowie die Anzahl an benötigten Fitnessfunktionsaufrufen.

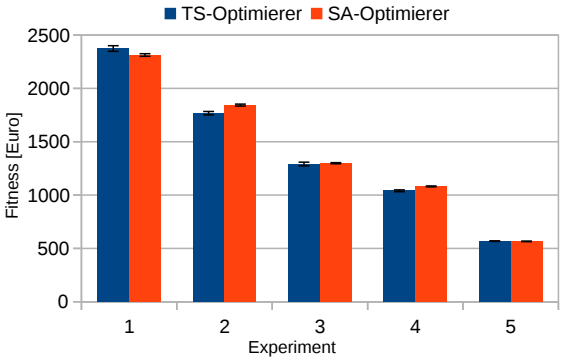
a) **Auswertung der Standardabweichung der Fitness**
In diesem Schritt wurde für jeden Optimierer die Standardabweichung der Fitnesswerte über die einzelnen Wiederholungen der Experimente aus Schritt 1 ausgewertet. Die Werte der Standardabweichung sind, bezogen auf die Fitness des jeweiligen Experiments, sehr gering (s. Abb. 46(a)). Die Fitness der Optimierer nimmt von Experiment zu Experiment ab, da die Experimente in dieser Abbildung nach absteigender Gesamtleistung des VK sortiert sind. Somit kann weniger Energie verkauft und folglich weniger Gewinn erzielt werden kann.

Um die Standardabweichungswerte noch genauer untersuchen zu können, wurden diese ins Verhältnis zur Fitness des in dem jeweiligen Experiment besten Optimierers gesetzt. Insgesamt wurde die Standardabweichung folgendermaßen normalisiert:

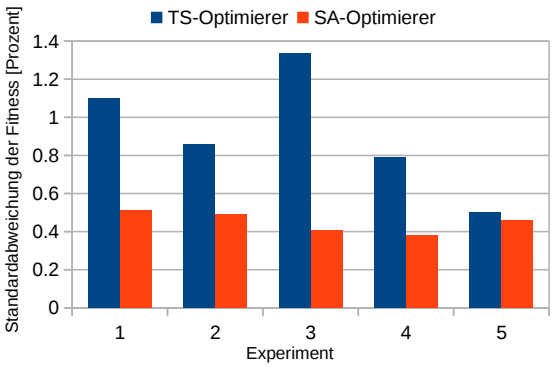
$$\frac{\text{Standardabweichung}}{\max\left(\text{Fitness SA-Optimierer, Fitness TS-Optimierer}\right)} \cdot 100$$

Abbildung 46(b) zeigt, dass der SA-Optimierer unabhängig vom verwendeten Dekodierer bei jedem Experiment eine geringere oder dieselbe relative Standardabweichung wie der TS-Optimierer besitzt. Der SA-Optimierer ist somit zuverlässiger als der TS-Optimierer. Da die relativen Standardabweichungen allerdings mit Werten von unter 2% sehr gering und im Realbetrieb eines VK vernachlässigbar sind, genügen diese Werte nicht, um den SA-Optimierer insgesamt als den besseren der beiden Optimerer auszuzeichnen. In den nachfolgenden Schritten wird aufgrund der geringen Standardabweichungswerte auf eine Wiederholung der Experimente verzichtet.

b) **Auswertung der Fitness in Abhängigkeit von der Laufzeit**
Es wurde untersucht, welche Auswirkung eine längere Laufzeit auf die Fitness hat. Dazu



(a) Fitness und absolute Standardabweichung



(b) relative Standardabweichung

Abbildung 46: Fitness und Standardabweichung mit euklidischem Dekodierer

wurde das bereits bei der Evaluation der inneren Optimierer verwendete Design (s. Abschnitt 6.10.5) mit jeder Optimierungsdauer $x \in \{2, 15, 35, 300\}$ (Angaben in Minuten) durchgeführt. Dieses Design deckt im Gegensatz zu dem bei der Auswahl des Dekodierers eingesetzen Design sehr viel größere Wertebereiche der Faktoren ab und umfasst eine größere Anzahl an Experimenten. Für die Auswertung wurde betrachtet, wie groß die Änderung der Fitness bei einer Ausführungsdauer von x Minuten zur Fitness bei einer Ausführungsdauer von 2 Minuten ist, wobei $x > 2$. Die Änderungswerte wurden für eine bessere Vergleichbarkeit zu Prozentangaben normiert:

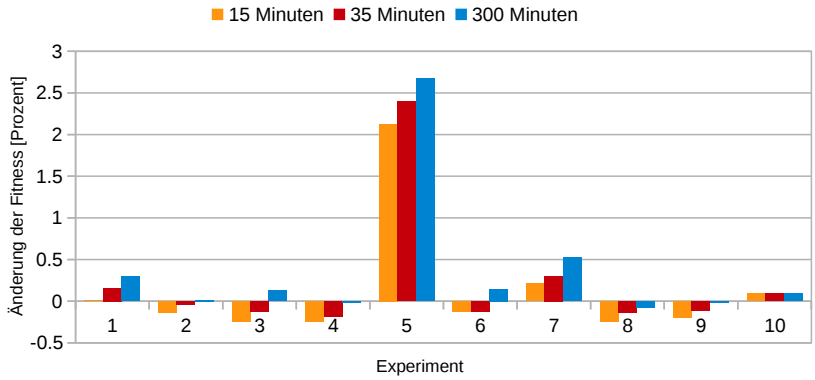
$$\frac{\text{Fitness des Optimierers bei x Minuten} \;-\; \text{Fitness des Optimierers bei 2 Minuten}}{\max\left(\text{Fitnesswerte des Optimierers bei 2, 15, 35, 300 Minuten}\right)} \cdot 100$$

Ein positiver Wert sagt aus, dass der Optimierer bei längerer Laufzeit als 2 Minuten eine größere Fitness erreichen konnte; ein negativer Wert, dass sich die Fitness verringert hat.

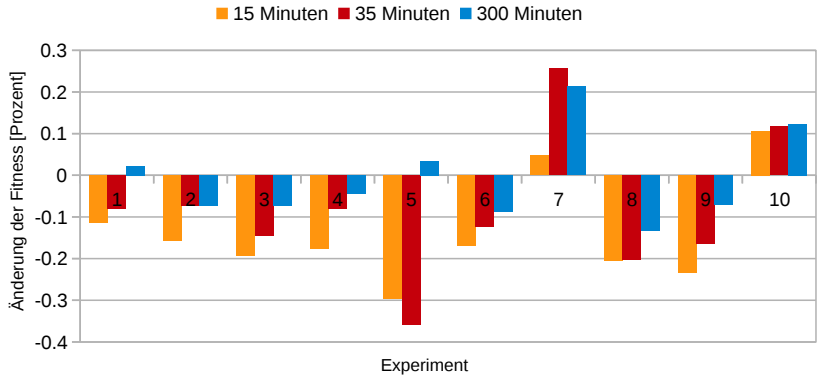
Abbildung 47 zeigt für den TS-Optimierer, dass sich die Fitness bei der Ausführungsdauer von 15 Minuten gegenüber der bei 2 Minuten bei 6 von 10 Experimenten verschlechtert. Wird die Ausführungsdauer weiter gesteigert, nehmen die Änderungswerte zu, d. h. die absolute Fitness vergrößert sich. Beim SA-Optimierer ist dieser Trend ebenfalls deutlich ausgeprägt. Bei beiden Optimierern fällt auf, dass Laufzeiten über 2 Minuten oft zu schlechteren Fitnesswerten führen: Selbst bei 300 Minuten erreicht der TS-Optimierer nur in 6 von 10 Experimenten eine bessere Fitness, der SA-Optimierer sogar nur bei 4 Experimenten. Insgesamt sind bei beiden Optimierern die Änderungen der Fitnesswerte, mit Ausnahme des auffallend abweichenden Experiments 5, mit Werten von $\pm$ 0,5% sehr gering. Daher wurde eine Optimierungsdauer von 2 Minuten als ausreichend angenommen und für die folgenden Experimente genutzt.

c) **Auswertung der Lösungsgüte**

Für dasselbe Design wurde außerdem untersucht, welcher der Optimierer generell die bessere Fitness besitzt. Abbildung 48 zeigt für die Ausführungsdauer von 2 Minuten, dass der TS-Optimierer nur zwischen 94% und 97% der Fitness des SA-Optimierers erreicht.



(a) TS-Optimierer



(b) SA-Optimierer

Abbildung 47: Änderung der Fitness bei unterschiedlichen Laufzeiten im Vergleich zur Fitness bei einer Laufzeit von 2 Minuten

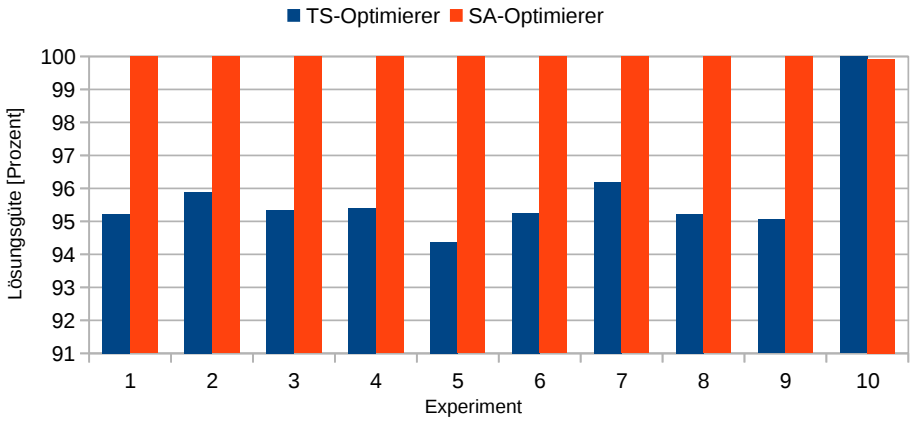


Abbildung 48: Lösungsgüte bei einer Laufzeit von 2 Minuten

Auch bei den höheren Optimierungsdauern erreicht der TS-Optimierer nie eine Lösungsgüte außerhalb dieses Bereichs. Eine Ausnahme ist das Experiment 10, bei welchem der TS-Optimierer bei jeder Ausführungszeit eine geringfügig bessere Fitness hat. Dieses ist allerdings ein Ausreißer, denn einerseits ist dieses Experiment zwar im Vergleich zu den übrigen eher klein - es gibt nur sehr wenig Anlagen mit relativ wenig Fahrplänen - und bei sehr kleinen Experimenten erreicht der TS-Optimierer eine ähnliche Fitness wie der SA-Optimierer (s. Schritt 3a). Andererseits ist dieses Experiment aber deutlich größer als die Experimente, die in Schritt 3a untersucht wurden und für den TS-Optimierer eine kleinere Fitness ergaben. Daher wäre auch bei Experiment 10 für den TS-Optimierer eine geringere Fitness als für den SA-Optimierer zu erwarten gewesen.

Wird die Differenz der Lösungsgüten der beiden Optimierer betrachtet

$$\text{Lösungsgüte SA-Optimierer [\%]} - \text{Lösungsgüte TS-Optimierer [\%]}$$

ist ersichtlich (s. Abb. 49), dass die Differenz tendenziell abnimmt, wenn den Optimierern mehr Zeit zur Verfügung gestellt wird. Dennoch erreicht der TS-Optimierer bei weitem nicht die Lösungsgüte des SA-Optimierers.

Grund hierfür ist vermutlich, dass die Mutation des TS-Optimierers weniger stark als die des SA-Optimierers ist und der TS-Optimierer folglich den Suchraum nicht so weit durchsuchen kann. In einem Mutationsschritt des SA-Optimierers werden alle Fahrpläne des Einsatzplans mutiert. Die Anzahl der mutierten Intervalle eines Fahrplans hängt von der Mutationsstärke ab. Diese wird nach Rechenbergs 1/5-Regel bestimmt. Beim TS-Optimierer wird nur ein Fahrplan mutiert, von diesem werden alle Intervalle geändert. Insgesamt werden so beim TS-Optimierer 96 Intervalle mutiert, während beim SA-Optimierer abhängig von der Anlagenanzahl und der Mutationsrate die aufsummierte Anzahl mutierter Intervalle deutlich darüber liegen kann. Bei 300 Anlagen sind dies beispielsweise mindestens 300 Intervalle. Somit kann der SA-Optimierer den Suchraum schneller und weiträumiger durchsuchen und wahrscheinlicher auf eine bessere Lösung stoßen.

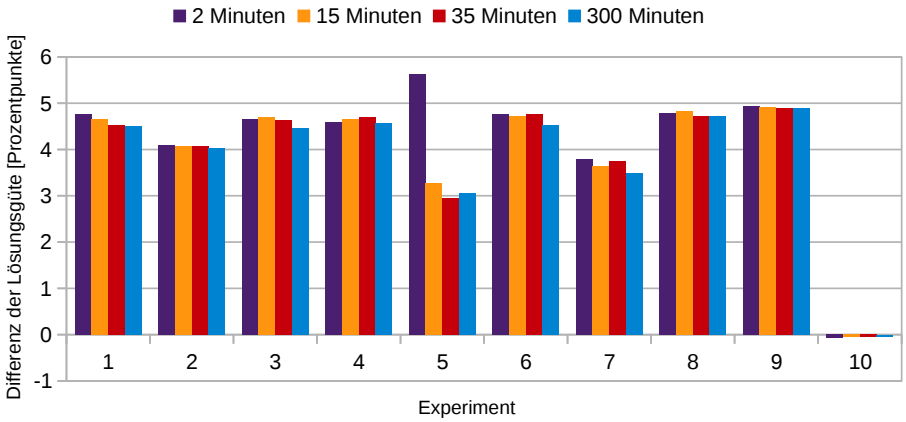


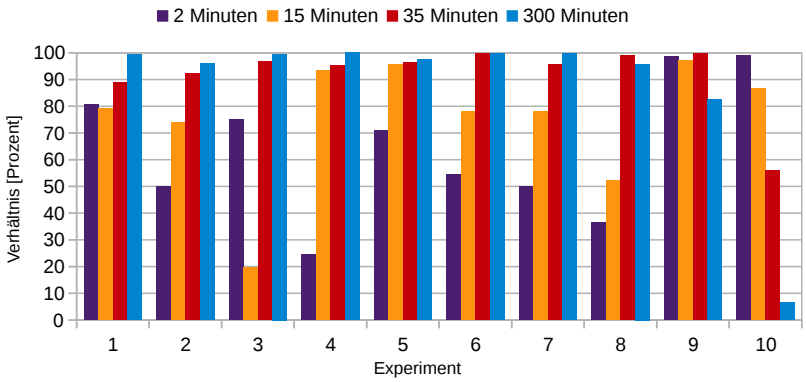
Abbildung 49: Differenz der Lösungsgüte des TS-Optimierers zum SA-Optimierer bei unterschiedlichen Laufzeiten

d) **Auswertung der Anzahl an Fitnessfunktionsaufrufen**
Weiterhin wurde das Verhältnis der Anzahl an Fitnessfunktionsaufrufen, nach welcher ein Optimierer die finale Lösung bestimmt hatte, zur Gesamtanzahl an Aufrufen untersucht. In fast allen Fällen war das Verhältnis beim TS-Optimierer höher als beim SA-Optimierer (s. Abb. 50). Dies bedeutet, dass der TS-Optimierer oft erst gegen Ende der Optimierung die endgültige Lösung findet. Wird dem TS-Optimierer mehr Laufzeit zur Verfügung gestellt, nimmt das Verhältnis der benötigten zur gesamten Anzahl zu: Bei 300 Minuten beträgt das Verhältnis in 8 von 10 Experimenten über 90%. Beim SA-Optimierer ist keine so eindeutige Entwicklung zu erkennen. In einigen Experimenten, wie beispielsweise den Experimenten 8, 9 und 10, ist eine Zunahme des Verhältnisses bei größerer Laufzeit zu erkennen, während in anderen Experimenten, wie in 1 und 7 bzw. 3, eine Abnahme bzw. geringe Änderung erfolgt.

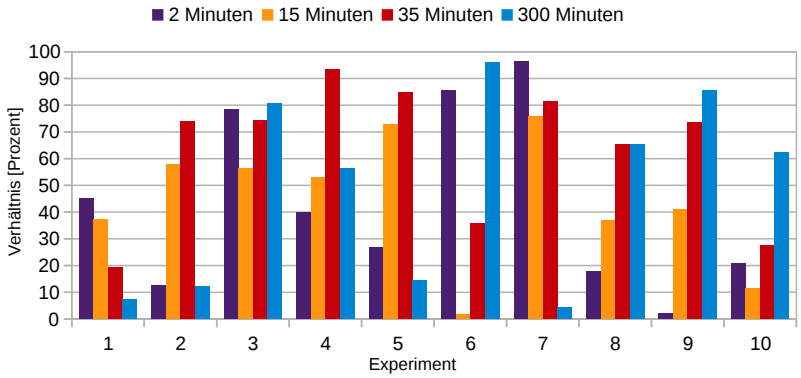
Diese Ergebnisse erklären die Befunde in Schritt 2b: Da der TS-Optimierer oft erst spät im Optimierungsverlauf die Endlösung erreicht, führen bei ihm Laufzeiten von 300 Minuten öfter zu besseren Ergebnissen als beim SA-Optimierer. Beim SA-Optimierer, welcher in knapp der Hälfte der Experimente weniger als die Hälfte der gesamten Fitnessfunktionsaufrufe benötigt, führen Laufzeiten von mehr als 2 Minuten in mehr als der Hälfte der Experimente zu schlechteren Fitnesswerten (s. Abb. 47), da der Optimierer bereits gefundene gute Lösungen zugunsten weiterer Durchsuchung des Suchraums aufgibt.

3. **Untersuchung ausgewählter Faktoren**
Während in den vorigen Evaluationsschritten ausschließlich Latin Hypercube Designs zum Einsatz kamen, wurden in den letzten beiden Schritten Designs verwendet, bei denen nur die Faktorstufen ausgewählter Faktoren variiert wurden, während die Faktorstufen der übrigen gleich blieben.

a) **Untersuchung der Fahrplananzahl**
Es wurde ein Design mit zehn Experimenten erstellt, wobei nur die Anzahl an Fahrplänen variiert wurde:



(a) TS-Optimierer



(b) SA-Optimierer

Abbildung 50: Verhältnis benötigter Fitnessfunktionsaufrufe zur Gesamtanzahl der Aufrufe

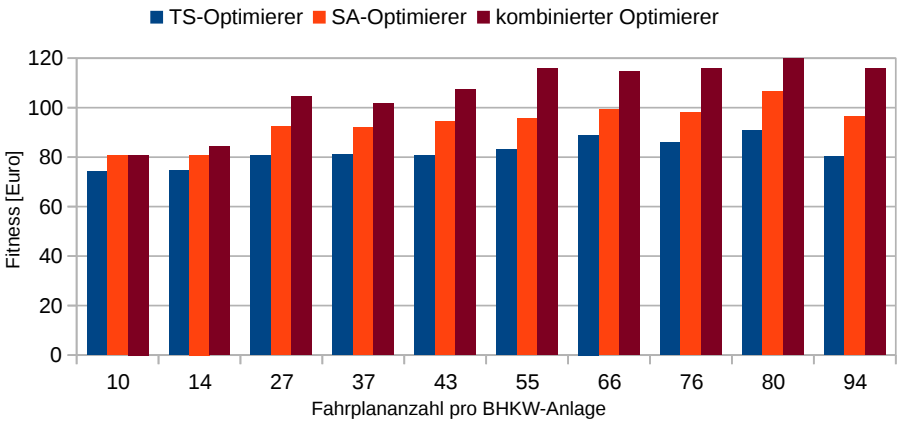


Abbildung 51: Fitness bei Änderung der Fahrplananzahl

Anzahl der Anlagen im VK	10
Gesamtleistung des VK	150 kWh
Anteil der BHKW-Leistung an der Gesamtleistung des VK	1,0
Anzahl an Fahrplänen pro BHKW-Anlage	$\in \{1;\ldots;100\}$

Die Gesamtleistung des VK wurde so gewählt, dass die Mindestgebotsmenge des Marktes (100 kWh) überschritten werden konnte. Dieses Design und das des nächsten Schrittes wurden im Vergleich zu den vorigen klein gewählt, da sie auch vom kombinierten Optimierer in akzeptabler Zeit ausgeführt werden sollten.

Aus den Ergebnissen dieses Evaluationsschrittes (s. Abb. 51) können vier Erkenntnisse gezogen werden. Zum einen ist ersichtlich, dass die Optimierer bessere Fitnesswerte erzielen, wenn mehr Fahrpläne pro BHKW-Anlage zur Verfügung stehen. Dies kann damit erklärt werden, dass bei vielen Fahrplänen mehr Möglichkeiten bestehen, die Fahrpläne so zu wählen, dass möglichst wenig nicht verkaufbare Energie erzeugt wird. Falls zu wenig Fahrpläne zur Verfügung stehen, können die Optimierer nicht auf solche Alternativen zurückgreifen.

Das zweite Ergebnis ist, dass der kombinierte lineare Optimierer in jedem Experiment bessere Ergebnisse liefert als die Heuristiken. Dieses Ergebnis war aufgrund der Eigenschaft linearer Optimierungsverfahren, immer die optimale Lösung zu berechen, zu erwarten. Bei größeren Experimenten besitzen allerdings die Heuristiken mit einiger Wahrscheinlichkeit eine höhere Fitness (Begründung s. Schritt 3b).

Ein weiteres Ergebnis ist, dass der TS-Optimierer auch bei diesen Experimenten eine geringere Fitness als der SA-Optimierer aufweist, obgleich diese Experimente im Vergleich zu denen aus Schritt 2c klein sind.

Außerdem ist in Abbildung 51 zu sehen, dass die Differenz zwischen den Fitnesswerten bei geringer Fahrplananzahl tendenziell kleiner ist als bei größerer Fahrplananzahl. Die Ursache für dieses Ergebnis ist, dass bei geringer Fahrplananzahl nicht viele Auswahl-

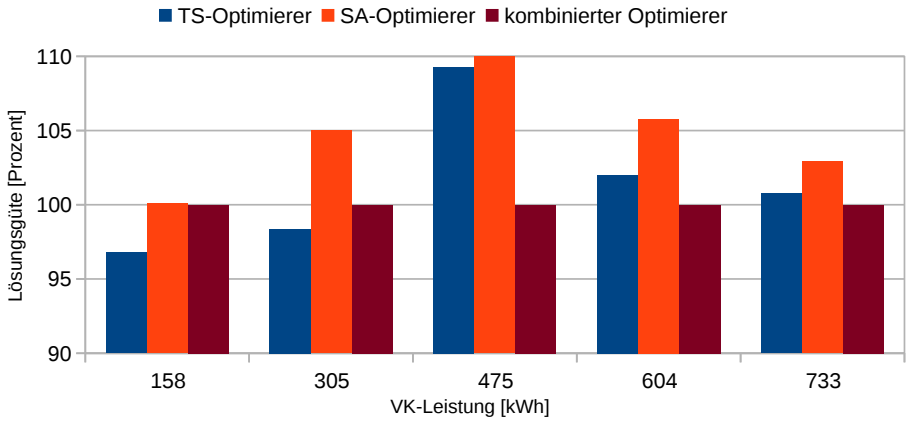


Abbildung 52: Lösungsgüte bei unterschiedlicher VK-Leistung (erstes Design)

möglichkeiten bestehen, sodass die Heuristiken den Suchraum gut durchsuchen und mit großer Wahrscheinlichkeit die beste Lösung finden können.

b) **Untersuchung der VK-Leistung und Anlagenanzahl**

Der letzte genauer untersuchte Faktor ist die Gesamtleistung des VK. Dafür wurden zwei Designs mit je 5 Experimenten erstellt. Beim ersten Design wurde nur die Gesamtleistung geändert:

Anzahl der Anlagen im VK	10
Gesamtleistung des VK	$\in \{150;\ldots;750\}$ kWh
Anteil der BHKW-Leistung an der Gesamtleistung des VK	1.0
Anzahl an Fahrplänen pro BHKW-Anlage	10

Beim zweiten wurde zusätzlich die Anlagenanzahl geändert. Beide Faktoren wurden dabei jeweils um denselben Faktor geändert, sodass über die Experimente hinweg ähnliche Anlagengrößen verwendet wurden:

Anzahl der Anlagen im VK	$\in \{10;\ldots;50\}$
Gesamtleistung des VK	$\in \{150;\ldots;750\}$ kWh
Anteil der BHKW-Leistung an der Gesamtleistung des VK	$1,0$
Anzahl an Fahrplänen pro BHKW-Anlage	10

Die Auswertung beider Designs ergab, dass auch hier der SA-Optimierer in jedem Experiment eine bessere Lösungsgüte als der TS-Optimierer besaß (Abbildungen 52 und 53). Außerdem ist erkennbar, dass der SA-Optimierer immer und der TS-Optimierer fast immer eine bessere Lösungsgüte als der kombinierte Optimierer besitzen.

Dies ist durch die zugunsten kürzerer Laufzeit vereinfachte Implementierung des kombinierten Optimierers erklärbar. Durch die Auswahl kontinuierlicher statt 100kWh-diskreter Volumina für die Produkte plant der kombinierte Optimierer unter Umständen Leistung

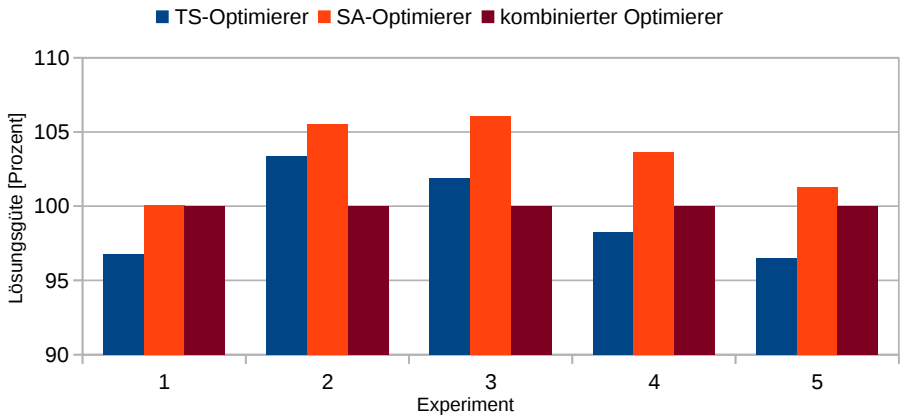


Abbildung 53: Lösungsgüte bei unterschiedlicher VK-Leistung und Anlagenzahl (zweites Design)

in Produkten ein, die aufgrund der Diskretisierungsregelung nicht verkauft werden kann (s. Abschnitt 6.9.4). Bei größeren Experimenten, insbesondere bei solchen mit größerer Gesamtleistung des VK, steigt die Wahrscheinlichkeit eines solchen Falls und somit der finanzielle Verlust. Die Heuristiken hingegen planen solche Leistung möglicherweise gewinnbringender in anderen Produkten ein.

Trotzdem befindet sich die Lösungsgüte der heuristischen Optimierer in der Nähe derjenigen des kombinierten Optimierers: Die Lösungsgüte des SA-Optimierers ist bis zu 10% besser als die des kombinierten Optimierers. Die des TS-Optimierers bewegt sich im Bereich $\pm$ 5% in Bezug auf den kombinierten.

Werden alle Evaluationsergebnisse der äußeren Optimierer insgesamt betrachtet, erweist sich der SA-Optimierer mit euklidischem Dekodierer und RI-Optimierer als innerem Optimierer als bester der heuristischen Optimierer. Er besitzt gegenüber dem TS-Optimierer eine höhere Zuverlässigkeit und eine bessere Lösungsgüte. Außerdem erreicht er eine höhere Lösungsgüte als der kombinierte Optimierer und zeichnet sich gegenüber diesem zusätzlich dadurch aus, dass für ihn, wie auch für den TS-Optimierer, eine kurze Laufzeit von 2 Minuten ausreichend ist. Aus diesem Grund

6.10.6. Implementierung

Das Evaluationsmodul ist der Rahmen, um Designs zu erstellen, zu speichern, auszuführen und die Ergebnisse zu sichern. Die verschiedenen Teile der Evaluation

- Erstellung des Designs
- Evaluation der inneren Optimierer
- Evaluation der äußeren Optimierer
- Ausführung des kombinierten Optimierers
- Ausführung des kombinierten Optimierers für mehrere Tage

können einzeln ausgewählt und zeitlich voneinander getrennt ausgeführt werden. Im Folgenden werden die wichtigsten Klassen vorgestellt, die diese Funktionalität bereitstellen. Weitere Erklärungen zur Ausführung der Evaluation und zu den vom Nutzer im Code einzustellenden Parametern sind im Anhang F.4 nachlesbar.

EvaluationMain Die Klasse `EvaluationMain` ist die zentrale Klasse des Evaluationsmoduls. Sie startet die RMI-Services für die Kommunikation mit den anderen Modulen, fragt den Nutzer nach seinem Evaluationsziel und führt dieses anschließend aus. Falls ein Design erstellt werden soll, stößt die `EvaluationMain` den `EvaluationParameterSetCreator` an, welcher die konkreten Wertebelegungen der Faktoren für die einzelnen Experimente bestimmt und abspeichert. Falls das Ausführen der Optimierer gewünscht ist, wird zunächst das abgespeicherte Design geladen und für jedes Experiment des Designs ein Optimierer mit den entsprechenden Daten erzeugt. Anschließend werden die Optimierer ausgeführt und die Evalulationsergebnisse berechnet und in einer Datei gespeichert.

EvaluationParameterSetCreator Ein `EvaluationParameterSet`-Objekt repräsentiert ein konkretes Evaluationsszenario: Es enthält die Wertebelegungen der Faktoren für ein Experiment. Der `EvaluationParameterSetCreator` ist dafür zuständig, alle `EvaluationParameterSet`-Objekte für ein Design zu erzeugen. Dafür muss der Nutzer im `EvaluationParameterSetCreator` die gewünschten Wertebereiche der Faktoren, den Designtyp (vollfaktoriell oder Latin Hypercube) und die Anzahl der Intervalle angeben, in welche die Wertebereiche eingeteilt werden sollen. Die Intervallanzahl bestimmt, abhängig vom ausgewählten Design, die Anzahl an Experimenten.

Für die Erstellung eines Designs teilt der `EvaluationParameterSetCreator` zunächst jeden Wertebereich in die angegebene Anzahl von Intervallen ein. Dann erstellt er die vom Designtyp vorgesehenen Intervallkombinationen und wählt aus jedem Intervall einen Wert zufällig aus. Anschließend wird mit Hilfe des `PlantConfigurationCreator` aus diesen Daten die Anlagenkonfiguration erstellt und in einer Textdatei gespeichert. Danach fragt der `EvaluationParameterSetCreator` die Marktdaten für den Evaluationstag vom Marktmodul und die Flexibilitäten für die Anlagen vom Flexibilitätenmodul ab. Zum Schluss erzeugt der `EvaluationParameterSetCreator` für die Evaluation der inneren (Produktportfolio-)Optimierer aus den Anlagenflexibilitäten einen Einsatzplan, indem er für jede Anlage jeweils den ersten Fahrplan aus der Fahrplanmenge auswählt. Diese Objekte (Werte der Faktoren, Anlagenkonfiguration, Marktdaten, Flexibilitäten, Einsatzplan) bilden zusammen das `EvaluationParameterSet` für ein Experiment. Abschließend wird jedes `EvaluationParameterSet` serialisiert und in einer Datei gespeichert, damit es zu einem späteren Zeitpunkt in der Evaluation verwendet werden kann. Wird der kombinierte Optimierer evaluiert, so wird dessen Fitness als optimale Lösung in dem entsprechenden deserialisierten `EvaluationParameterSet` gesetzt. Anschließend wird das `EvaluationParameterSet` wieder serialisiert, sodass für die Auswertung der äußeren Optimierer die Fitness des kombinierten Optimierers zur Verfügung steht.

PlantConfigurationCreator Die Aufgabe des `PlantConfigurationCreator` ist, ausgehend von den gegebenen Wertebelegungen der Faktoren

- Anzahl der Anlagen im VK

- Gesamtleistung des VK
- Anteil der Leistung der BHKW-Anlagen an der Gesamtleistung

eine konkrete Auswahl an parametrisierten Anlagentypen zu treffen (Anlagenkonfiguration). Diese Auswahl hätte auch über das Latin Hypercube Sampling realisiert werden können, wenn für jeden parametrisierten Anlagentyp ein Faktor eingeführt worden wäre, der die Anzahl solcher Anlagen repräsentiert. Dies wurde nicht umgesetzt, da es leichter war, die Qualität der Optimierer über abstraktere Faktoren, wie Anlagenanzahl und Gesamtleistung des VK (s. Abschnitt 6.10.3), zu untersuchen, als über die feingranularen Faktoren parametrisierter Anlagentypen, aus denen dann die abstrakteren Faktoren hätten abgeleitet werden müssen.

Es kann der Fall eintreten, dass es nicht möglich ist, eine Anlagenkonfiguration mit den im VPP-Modul hinterlegten parametrisierten Anlagentypen (6.4.1) so zu erstellen, dass die Werte aller obiger Faktoren exakt eingehalten werden. Da die Anlagenanzahl als relevant erachtet wurde, wird diese trotzdem in jedem Fall eingehalten. Die Gesamtleistung des VK und der Anteil der BHKW-Leistung werden so nah wie möglich angenähert.

Für die Erstellung der Anlagenkonfiguration rechnet der `PlantConfigurationCreator` zunächst den Leistungsanteil für jeden Anlagentyp in absolute Angaben um. Anschließend wählt er für jeden Anlagentyp zunächst ohne Berücksichtigung der Anlagenanzahl solange parametrisierte Anlagentypen aus, bis die gewünschte Leistung des Anlagentyps erreicht ist. Dabei wählt er zuerst den parametrisierten Anlagentyp mit der geringsten Maximalleistung aus und anschließend den nächstgrößeren, falls dieser die Gesamtleistung für diesen Anlagentyp nicht überschreitet usw., bis entweder

- der größte parametrisierte Anlagentyp erreicht ist oder
- der nächstgrößere parametrisierte Anlagentyp nicht mehr gewählt werden kann, da er die Gesamtleistung überschreiten würde.

In beiden Fällen wird wieder mit dem kleinsten parametrisierten Anlagentyp begonnen. Die Auswahl endet dann, wenn auch der kleinste parametrisierte Anlagentyp nicht ausgewählt werden kann. Anschließend wählt der `PlantConfigurationCreator` für den nächsten Anlagentyp die parametrisierten Typen aus. Dieses Vorgehen hat das Ziel, dass möglichst jede Anlagengröße berücksichtigt wird, aber kleine Anlagen bevorzugt werden. Kleine Anlagen werden bevorzugt, da reale virtuelle Kraftwerke vermutlich viele kleine Anlagen von Privathaushalten und einige wenige große Anlagen enthalten.

Im nächsten Schritt wird die Auswahl der Anlagen so korrigiert, dass die gewünschte Gesamtanlagenanzahl des VK exakt und die Gesamtleistung des VK möglichst nah erreicht wird. Dazu wird der Betrag der zu viel bzw. zu wenig ausgewählten Anlagenanzahl $diff$ folgendermaßen auf die Anlagentypen umgelegt:

$$\text{zu korrigierende Anzahl PV-Anlagen} = \frac{\text{Anzahl PV-Anlagen}}{\text{Gesamtzahl ausgewählter Anlagen}} * diff \tag{24}$$

$$\text{zu korrigierende Anzahl BHKW-Anlagen} = diff - \text{zu korrigierende Anzahl PV-Anlagen} \tag{25}$$

	B	C	D
1	ParameterSets\Optimizer	TabuSearchOptimizer	SimulatedAnnealingOptimize
2	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_AvergeAbsoluteFitness	74.3490185776	80.7858881555
3	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_SolutionQuality	0.9194415482	0.9990434776
4	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_ComputingTime	120.043	120.564
5	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_UntilEndFitnessFunction	96291	746712
6	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_UntilNowFitnessFunction	1245	76157
7	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_StandardDeviationOfFitn	0	0
8	ParameterSet_MaxPow-150,00_PlantCount-10_PowPerc-1,00_SchedCount-10_IsFeasible	true	true

Abbildung 54: Ausschnitt aus der Ausgabedatei der Evaluationsergebnisse

Anschließend wird die Anlagenanzahl für jeden Anlagentyp einzeln korrigiert. Falls x Anlagen zu viel ausgewählt worden waren, werden $x+1$ Anlagen des kleinsten parametrisierten Anlagentyps aus der Auswahl entfernt und stattdessen eine Anlage des parametrisierten Anlagentyps ausgewählt, deren Maximalleistung am nächsten an der aggregierten Leistung der entfernten Anlagen entspricht. Sind nicht genügend Anlagen des kleinsten parametrisierten Anlagentyps vorhanden, werden nacheinander so viele Anlagen der nächstgrößeren Typen entfernt, bis die Anzahl $x+1$ erreicht ist. Aufgrund dieses Vorgehens können entgegen der Zielsetzung des vorigen Schrittes Anlagenkonfigurationen entstehen, welche keine Anlagen des kleinsten Typs enthalten.

Wurden insgesamt zu wenig Anlagen ausgewählt, wird die Anlage entfernt, deren Maximalleistung am nächsten an der aggregierten Leistung des kleinsten parametrisierten Anlagentyps liegt. Statt dieser Anlage werden $x+1$ Anlagen des kleinsten parametrisierten Anlagentyps zusätzlich in die Anlagenkonfiguration aufgenommen.

OptimizerQuality Die Klasse `OptimizerQuality` bildet die Datenstruktur, um für einen Optimierer die Optimierungsergebnisse zu speichern und die Qualitätskriterien (s. Abschnitt 6.10.2) auszuwerten. Dafür implementiert sie die Durchschnittsbildung der Optimierungsergebnisse über die Wiederholungen der Experimente, die Berechnung der Standardabweichung der Fitnesswerte und die Bestimmung der Lösungsgüte.

EvaluationResult Die Klasse `EvaluationResult` wird verwendet, um die Evaluationsergebnisse eines Optimierertyps (innerer/äußerer/kombinierter) zusammenzufassen. `EvaluationResult` ist eine `HashMap`, welche für jeden Optimierer eines Typs für jedes `EvaluationParameterSet` das `OptimizerQuality`-Objekt enthält: `HashMap<OptimizerDescription, HashMap<EvaluationParameterSet, OptimizerQuality»`

Nachdem alle Optimierer eines Typs evaluiert wurden, schreibt die `EvaluationMain` das `EvaluationResult` mit Hilfe der Klasse `ResourceUtil` in eine csv-Datei. Ein Ausschnitt einer so erzeugten Datei ist in Abbildung 54 zu sehen. In der Datei sind die Qualitätskriterien für jedes `EvaluationParameterSet` untereinander angeordnet. Die Abbildung zeigt den Ausschnitt für nur ein `EvaluationParameterSet`. Für die Optimierer gibt es jeweils eine Spalte. In der Abbildung ist beispielsweise zu erkennen, dass der TS-Optimierer für das angegebene `EvaluationParameterSet` eine Fitness von 74,34€ erreicht hat. Mit Hilfe dieser Datei kann die Auswertung der Evaluationsergebnisse vereinfacht werden, da die Ergebnisse leicht grafisch veranschaulicht werden können.

6.11. GUI

Um die Benutzbarkeit des Systems zu gewährleisten, soll eine Benutzerschnittstelle implementiert werden. Über diese soll sowohl die Zusammenstellung des virtuellen Kraftwerks als auch das Datum für die Optimierung eingestellt werden können. Dabei wurde nicht vorgeschrieben, ob eine einfache Kommandozeilenschnittstelle oder eine grafische Anwendung umgesetzt werden soll. Die Entscheidung fiel aus ästhetischen Gründen auf eine grafische Benutzerschnittstelle. Ein weitere Grund ist die Möglichkeit, Optimierungsergebnisse grafisch aufbereiten und anzeigen lassen zu können. Da allerdings der Fokus der Projektgruppe auf anderen Aspekten liegt, soll die grafische Benutzerschnittstelle lediglich rudimentär umgesetzt werden. Aus diesem Grund werden im folgenden Abschnitt, in dem zunächst mögliche Technologien evaluiert und anschließend das angestrebte Design vorgestellt wird, nur die wichtigsten Aspekte beschrieben.

6.11.1. Technologien

Die Implementation einer grafischen Benutzerschnittstelle steht zunächst vor der grundsätzlichen Frage, ob diese als Benutzerprogramm oder als Webanwendungen umgesetzt werden soll. Für beide Optionen existieren zahlreiche Frameworks. Mögliche Frameworks für Webanwendungen sind AngularJS[12], JWT[13] und Vaadin[14], für Offline-Anwendungen existieren Java AWT und Swing Nachfolger JavaFX[15] und das Qt-Framework[16].

Die Umsetzung als Webanwendung hat den Vorteil, dass mit geringem Aufwand ansprechende Designs und Diagramme erzeugt werden können. Der Nachteil ist, dass gegebenenfalls zusätzliche Aspekte wie z. B. Zugriffsschutz und die Kommunikation mit dem Hauptprogramm betrachtet werden müssen. Für die Implementation als lokales Benutzerprogramm spricht, dass kein Webbrowser zum Ausführen benötigt wird und die Schnittstelle direkt in das Hauptprogramm integriert werden kann. Die Wahl fiel hierbei auf die Umsetzung als Benutzerprogramm.

Weitere Entscheidungskriterien sind die Möglichkeiten, nur in Java programmieren und ansprechende Diagramme generieren lassen zu können. Da Qt größtenteils in C++ geschrieben ist, fiel die Wahl auf JavaFX.

6.11.2. JavaFX

JavaFX [Ora] ist ein von Oracle entwickeltes Framework für plattformübergreifende Rich-Client Anwendungen und stellt den geistigen Nachfolger von Java AWT und Java Swing dar. JavaFX lässt sich vollständig über die Java API umsetzen und unterstützt sowohl JavaScript als auch Swing und SWT. Alternativ lässt sie die Benutzerschnittstelle auch über CSS und/oder spezielle XML-Dateien (sogenannte FXML-Dateien) gestalten. Auf diese Weise lassen sich Konzepte wie Model-View-Presenter umsetzen.

[12] http://angularjs.org/
[13] http://www.webtoolkit.eu/jwt
[14] http://vaadin.com/demo
[15] http://docs.oracle.com/javase/8/javase-clienttechnologies.htm
[16] http://www.qt.io/

6.11.3. Anforderungen an die Benutzerschnittstelle

Die Benutzerschnittstelle soll drei Funktionalitäten bereitstellen. Erstens soll die Zusammenstellung des virtuellen Kraftwerks festgelegt werden können. Dabei kann der Benutzer aus dem Pool der möglichen Anlagen die Anlagen und Anzahl seiner Wahl hinzufügen. Zweitens soll das Datum des Tages, für den die Optimierung durchgeführt werden soll, ausgewählt und die Optimierung anschließend gestartet werden können. Drittens sollen die Optimierungsergebnisse angezeigt werden können. Zu diesen Optimierungsergebnissen gehören das Produktportfolio, die Marktprognose, der Einsatzplan und die Fahrpläne der einzelnen Anlagen. Darüber hinaus sollen die Optimierungsergebnisse zusätzlich als Ausgabedateien erstellt werden, sodass der Benutzer die Möglichkeit hat andere Programme zur Auswertung der Ergebnisse verwenden zu können.

6.11.4. Implementierung

Die Benutzerschnittstelle wurde grob nach dem Model-View-Presenter-Konzept entworfen. Für jede der drei Hauptfunktionalitäten wird eine eigene View erstellt, die mittels FXML-Dateien gestaltet werden. In Abbildung 55 ist die Optimierungsansicht zu sehen. Hier kann der Benutzer seine Anlagenkonfiguration auf der linken Seite einsehen, während auf der rechten Seite Einstellungen vorgenommen werden können. Zu diesen Einstellungen gehört z. B. das Datum oder die Zeit, die für die Optimierung zur Verfügung steht.

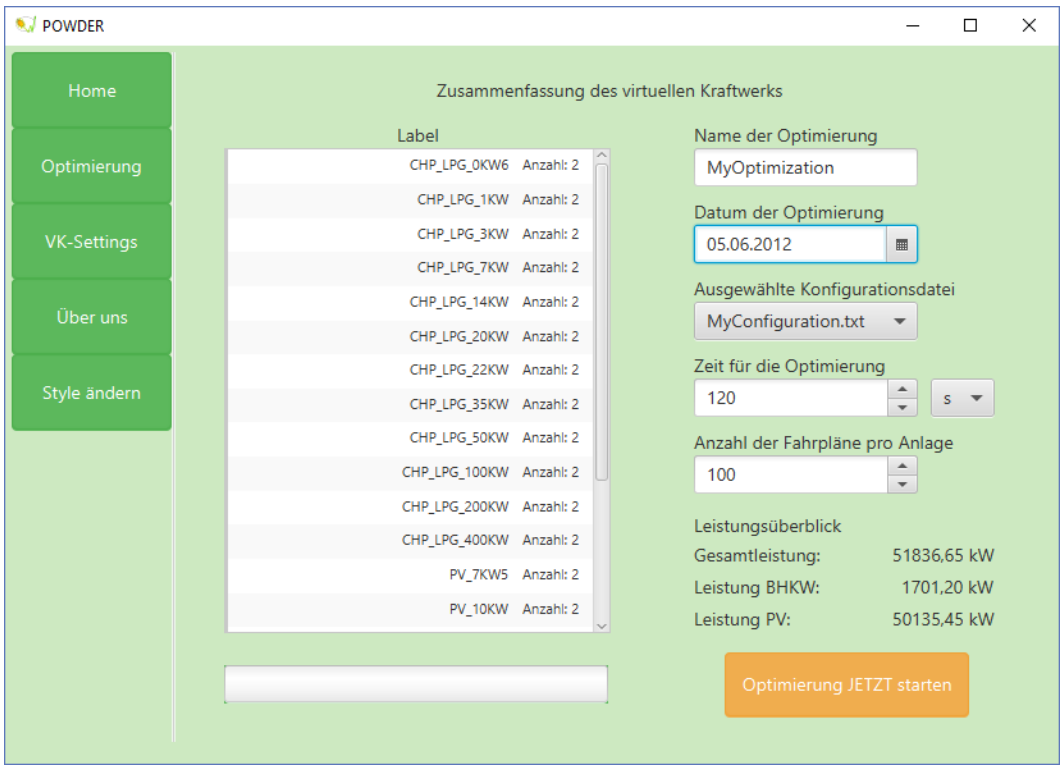


Abbildung 55: Optimierungsansicht der grafischen Benutzerschnittstelle

In Abbildung 56 ist die Ergebnisansicht zu sehen. Das angezeigte Diagramm ist in diesem Beispiel der Einsatzplan für eine bereits durchgeführte Optimierung. Außerdem ist der zu erwartende Gewinn und die erzeugte bzw. verkaufte Energie zu sehen.

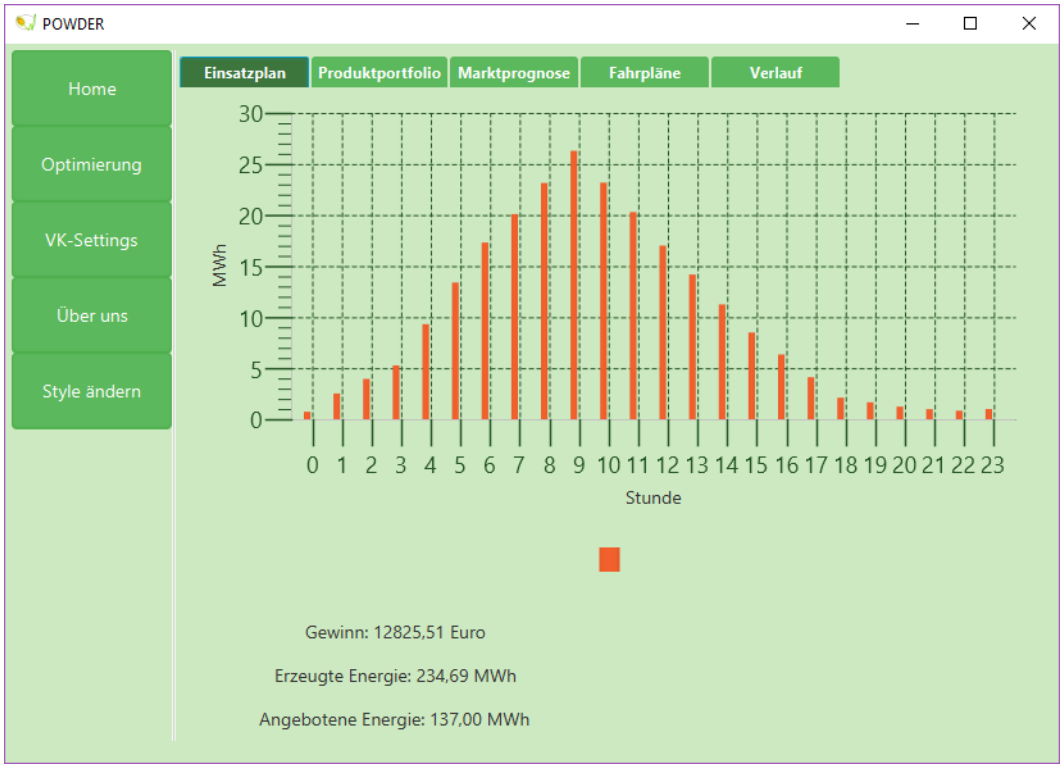


Abbildung 56: Optimierungsansicht der grafischen Benutzerschnittstelle

6.12. Gesamtablauf

In diesem Abschnitt soll in einem Gesamtablauf das Zusammenspiel der einzelnen Module beschrieben werden. Beim Start von *Powder* werden noch vor dem Laden der GUI im VPP-Modul (Abschnitt 6.4) die RMI-Services (s. Abschnitt 6.2.1) gestartet. In der GUI (s. Abschnitt 6.11) werden Einstellungen, wie das Startdatum und die Anlagenkonfiguration festgelegt. Sobald der Optimierungsprozess über die GUI initiiert wurde, wird der OPC-Server gestartet (s. Abschnitt 6.2.3). Außerdem wird das Startdatum an die einzelnen Modulservices weitergeleitet. Als nächstes wird eine `OptimizerDescription` erstellt, Flexibilitäten ermittelt und die Marktprognose bezogen.

Im Flexibilitätenmodul (s. Abschnitt 6.8) wird die Wettervorhersage vom Wettermodul (s. Abschnitt 6.6) bezogen und anschließend werden Anlagenmodelle instanziiert. Diese Anlagenmodelle werden in einer Simulationsumgebung ausgeführt, wodurch Fahrpläne generiert werden. Zum Schluss werden alle Fahrpläne aggregiert und als Flexibilitäten zurück an das VPP-Modul gesendet. Innerhalb des Marktmoduls (Abschnitt 6.7) werden historische Marktdaten und Wettervorhersagen bezogen. Auf diesen Daten wird mittels maschinellen Lernens eine Zeitreihenanalyse durchgeführt und das Ergebnis als Marktprognose zurückgeliefert. Flexibilitäten, Marktprognose und die `OptimizerDescription` werden an das Optimierungsmodul (Abschnitt 6.9) gesendet. Dort wird zunächst die `OptimizerDescription` ausgewertet und daraus der äußere und der innere Optimierer zusammen-

gebaut. Dabei erhält der äußere Optimierer, der in *Powder* immer den Einsatzplan optimiert, die Flexibilitäten und der innere Optimierer, der für das Produktportfolio zuständig ist, die Marktprognose. Anschließend wird der Optimierungsprozess gestartet.

Vereinfacht beschrieben, stellt der äußere Optimierer einen Einsatzplan zusammen, der dann an den inneren Optimierer gegeben wird. Der innere Optimierer erstellt zu dem Einsatzplan ein Produktportfolio und liefert es zurück. Anschließend wird dessen Lösungsqualität berechnet und als Fitnesswert dargestellt. Der Optimierungsprozess wird fortgeführt, bis die verfügbare Zeit abgelaufen ist. Die beste bis dahin gefundene Lösung wird als `OptimizationResult` zurück an das VPP-Modul gesendet. Dort werden der Einsatzplan und das Produktportfolio zunächst in CSV-Dateien geschrieben und anschließend in der GUI dargestellt.

7. Tests

In der Entwicklungsphase von *Powder* wurde stark auf Softwarequalität geachtet. Um diese kontinuierlich gewährleisten zu können, wurden viele Tests geschrieben. An zentralen Stellen halfen die Tests zur Sicherstellung bestehender Funktionalitäten, sodass Änderungen im Code darauf keine Auswirkungen hatten. Mit Hilfe von abstrakten Testfällen konnten grundlegende Anforderungen für alle konkreten Implementierungen getestet werden. So wurde beispielsweise bei allen Optimierern unabhängig von der Lösungsqualität getestet, ob valide Ergebnisse berechnet wurden.

Die folgenden Unterkapitel stellen das verwendete Testframework TestNG und die Mocking-Werkzeuge Mockito und PowerMockito vor und erläutern deren Funktionsweisen. Zum Schluss wird die Testabdeckung des Quellcodes dargestellt.

7.1. TestNG

TestNG[17] ist, wie JUnit, ein Java-Testframework, mit dessen Hilfe Codeabschnitte getestet werden können. So kann sichergestellt werden, dass die Softwarequalität eines Programms während der Entwicklung auf einem konstanten Level bleibt und weiter steigt. Auch das erwartete Verhalten von Methoden, Klassen und ganzen Modulen kann nach Änderungen im Quellcode durch Tests gewährleistet werden.

Für *Powder* wurde das Testframework TestNG gewählt, da Maven alle vorhandenen TestNG-Tests ohne weitere Konfigurationen erkennt und nach dem kompilieren ausführt. Darüber hinaus bietet TestNG im Gegensatz zu JUnit die Möglichkeit zur Umsetzung von Testabhängigkeiten. Tests, die von anderen Tests abhängig sind, werden nur ausgeführt, wenn die anderen Tests erfolgreich ausgeführt wurden. Dies hat den Vorteil, dass beim Testdurchlauf keine unnötigen Tests durchgeführt werden, bei denen bereits feststeht, dass sie nicht erfolgreich durchlaufen werden. Zudem sinkt die Testdauer bei einem fehlgeschlagenen Testdurchlauf aller Tests. Dies macht vor allem bei Tests Sinn, die einen größeren Zeitraum in Anspruch nehmen. Einen weiteren Vorteil bietet die Möglichkeit, Tests zu parametrisieren. So können ähnliche Testfälle mit verschiedenen Parametern ausgeführt werden, ohne zusätzliche, ähnliche Tests schreiben zu müssen.

In vielen Testfällen wurden die Mocking-Werkzeuge Mockito und PowerMockito verwendet, auf die folgend eingegangen wird.

7.2. Mockito

Tests von Funktionalitäten können schnell auf das Problem stoßen, dass die zu testende Funktionalität andere komplexe Module aufruft oder eine Verbindung zur Datenbank benötigt. Somit ist das Testergebnis davon abhängig, dass die benutzten komplexen Module und externen Verbindungen funktionstüchtig sind. Damit die zu testende Funktionalität möglichst gekapselt und unabhängig von anderen Modulen oder Verbindungen zu externen Schnittstellen getestet werden kann, bieten verschiedene Mocking-Tools die Möglichkeit, Bestandteile der zu testenden Funktionalität zu mocken. Der Aus-

[17] http://testng.org/doc/documentation-main.html

druck „Mocken“ ist vom englischen Adjektiv *mock* (= schein, pseudo) abgeleitet und verdeutlicht das Verwenden von Pseudo-Schnittstellen.

Für Java gibt es gibt diverse, frei verfügbare Mocking-Werkzeuge, wie zum Beispiel Mockito[18], EasyMock[19] und jMockit[20]. Für die Tests in *Powder* wurde Mockito verwendet, da EasyMock und jMockit im Vergleich weniger intuitiv zu bedienen sind. Im Code-Ausschnitt 8 wird verdeutlicht, wie ein einfacher Mock eines komplexen Services in einem zu testenden Service verwendet wird. In diesem einfachen Beispiel soll die Methode *methodThatUsesComplexService()* der Klasse *ServiceToTest* getestet werden. Die Klasse *ServiceToTest* ist im Code-Ausschnitt 7 angegeben.

Code-Ausschnitt 7: ServiceToTest

```
1  public class ServiceToTest {
2    private ComplexService complexService;
3
4    public String methodThatUsesComplexService() {
5      return "complexService returned: " + complexService.doComplexThings();
6    }
7
8    public void setComplexService(ComplexService complexService) {
9      this.complexService = complexService;
10   }
11 }
```

Der hier zu sehende *ComplexService* soll gemockt werden, um für diesen Test irrelevante und komplexe Berechnungen zu vermeiden. Mit Mockito kann die Testmethode folgendermaßen aussehen:

Code-Ausschnitt 8: Testmethode für ServiceToTest

```
1  @Test
2  public void mockTest() {
3    // Erstelle einen Mock der Klasse ComplexService
4    ComplexService complexService = Mockito.mock(ComplexService.class);
5    // Gebe den String "complexResult" zurück, wenn die Methode
6    // complexService.doComplexThings() aufgerufen wird
7    Mockito.when(complexService.doComplexThings()).thenReturn("complexResult");
8    // Erstelle das zu testende Objekt
9    ServiceToTest serviceToTest = new ServiceToTest();
10   // Setze den gemockten Service im serviceToTest
11   serviceToTest.setComplexService(complexService);
12   // Evaluiere das Ergebnis von serviceToTest.methodThatUsesComplexService()
13   String result = serviceToTest.methodThatUsesComplexService();
14   assertEquals(result, "complexService returned: complexResult");
15 }
```

[18] http://mockito.org/
[19] http://easymock.org/
[20] http://jmockit.org/

Die Möglichkeiten von Mockito beschränken sich darauf, Methoden zu mocken, die *protected* oder *public*, nicht *final* und nicht *statisch* sind. Um darüber hinaus private, finale oder statische Methoden zu mocken, kann das auf Mockito aufbauende Mocking-Tool PowerMockito verwendet werden.

7.3. PowerMockito

Neben Mockito gibt es mächtigere Mocking-Tools, welche das Mocken von *privaten*, *finalen* und *statischen* Methoden ermöglichen. Dadurch können Testfälle einfacher geschrieben werden, ohne den produktiven Quellcode anpassen zu müssen. Beispiele für mächtigere Mocking-Tools sind jMockit und PowerMockito[21]. Für Powder wurde das auf Mockito aufbauende Mocking-Tool PowerMockito verwendet, da die Handhabung sehr ähnlich zu Mockito ist. Dadurch vereinfacht sich die Einarbeitung in PowerMockito um ein Vielfaches. Durch einen eigenen Java Classloader und Bytecode Manipulationen ermöglicht PowerMockito zusätzlich das Mocken von Konstruktoren mit beliebigem Modifikator.

7.4. Testabdeckung

Durch die testgetriebene Softwareentwicklung in *Powder* wurden wichtige Bestandteile des Quellcodes durch Testfälle abgedeckt. Die Testabdeckung ist in den drei Granularitäten *Klassen*, *Methoden* und *Zeilen* zu unterteilen. Jede Granularität gibt an, welche prozentualen Anteile aller Klassen, Methoden oder Zeilen abgedeckt werden, wenn alle Testfälle durchgeführt werden. Nachdem alle Tests in *Powder* durchgelaufen sind, ergeben sich folgende Testabdeckungen:

Klassen	Methoden	Zeilen
53%	42%	38%

Tabelle 6: Testabdeckung des gesamten Quellcodes in *Powder*

Die höchste Testabdeckung liegt im Optimierungsmodul vor, da hier essentielle Algorithmen implementiert wurden, bei denen eine hohe Testabdeckung nötig war:

Klassen	Methoden	Zeilen
79%	77%	73%

Tabelle 7: Testabdeckung des Optimierungsmoduls

[21] https://github.com/jayway/powermock#documentation

8. Fazit

In diesem Kapitel wird zunächst sowohl methodisch, als auch inhaltlich auf das Projekt zurückgeblickt und anschließend ein Ausblick auf ein mögliches Vorgehen bei einer zukünftigen Weiterentwicklung gegeben.

8.1. Reflexion

Diese Reflexion ist in zwei Teile gegliedert. Zunächst wird das Vorgehen innerhalb der Projektgruppe reflektiert und anschließend dargestellt, inwieweit die Anforderungen an *Powder* erfüllt wurden.

Rückblick auf die Projektgruppe Bei Anwesenheit aller Projektgruppenmitglieder wurde eine Reflexion über das gemeinsame Vorgehen innerhalb der Projektgruppe durchgeführt. Dabei wurden positive und negative Aspekte gesammelt, die im Folgenden aggregiert und gegebenenfalls verallgemeinert dargestellt werden.

Besonders hervorzuheben ist der Zusammenhalt innerhalb der Gruppe und die Arbeitsbereitschaft der einzelnen Gruppenmitglieder. Das hatte zur Folge, dass sich selbst für unbeliebte Aufgaben immer Freiwillige gemeldet haben. Die gemeinsamen Sitzungen liefen in der Regel konzentriert ab und Unpünktlichkeit war eher die Ausnahme. Dies lag zum Teil sicher auch an der *Kuchenpunkteregelung*, die von allen Mitgliedern positiv aufgenommen wurde.

An der Entwicklung der Gruppe haben auch die häufigen Retrospektiven und Evaluationen des Vorgehens einen großen Anteil. Nach jeder Gruppensitzung wurde diese reflektiert und aus den positiven und negativen Anmerkungen Verbesserungen für zukünftige Sitzungen geschlussfolgert und diese als Konventionen und Richtlinien festgehalten. Eine dieser Verbesserungen war die Verkürzung der Sprintdauer von anfangs zwei Wochen auf eine Woche. Wurde bei zweiwöchigen Sprints in der ersten Woche oft wenig getan, konnte bei den einwöchigen Sprints jede Woche effektiv ausgenutzt werden. Gleichzeitig war die Menge der Aufgaben für den aktuellen Sprint überschaubarer. Zusätzlich haben die einwöchigen Sprints einen Motivationszuwachs bewirkt.

Als weiterer positiver Aspekt wurde die letzte Projektphase benannt. In dieser Phase wurden für die gesamte Gruppe feste, gemeinsame Arbeitszeiten festgelegt. Die wichtigsten Vorteile dieses Vorgehens waren die Verbesserung der Kommunikation der Mitglieder untereinander (Verantwortliche konnten sofort um Hilfe gebeten werden) und der Wegfall des meist komplizierten Kleingruppenzeitmanagements.

Im Verlauf des Projekts sind ebenfalls Schwierigkeiten und Hindernisse aufgetreten. Das erste Problem trat hauptsächlich in der zweiten Hälfte der Projektlaufzeit, während der Implementierungsphase auf. Einige Aufgaben wurden nicht vollständig oder mit unzureichender Güte erledigt und trotzdem als abgeschlossen markiert. Hinzu kam, dass bei darauf aufbauenden Aufgaben die Bearbeiter darauf vertraut haben, dass die vorangegangene Aufgabe vollständig bzw. mit ausreichender Güte erledigt wurde. Dies hat die nachträgliche Fehlersuche immens erschwert. Viele daraus resultierende Fehler sind sehr lange unentdeckt geblieben, manche traten erst kurz vor Ende auf. Teilaufgaben, die zeitlich nicht mehr fertig gestellt werden konnten, wurden oft lediglich innerhalb eines Code-Kommentars vermerkt. Dem hätte entgegen gewirkt werden können, wenn solche *TODOs* auch als Aufgaben in

das Aufgabenmanagementwerkzeug JIRA eingetragen worden wären. Gründe hierfür waren oft Zeitmangel, der unter anderem dadurch entstand, dass bei einigen Aufgaben zuvor noch viel andere Arbeit erledigt werden musste, beispielsweise wenn Fehler auftraten, die bisher noch nicht aufgetreten waren. Viele Fehler wurden auch deshalb nicht sofort entdeckt, da es erst sehr spät im Projektverlauf die Möglichkeit gab, die Software vollständig auszuführen. Ein Framework zum Ausführen, wie das Evaluationsmodul, wäre zu Beginn der Implementierungsphase sehr hilfreich gewesen.

Die Entscheidung, bei der Analyse- und Architekturphase nach SGAM vorzugehen, beurteilt die Projektgruppe im Nachhinein mit gemischten Meinungen. Hilfreich war, dass durch die Modellierung des Business- und des Function Layers in der SGAM-Analysephase die Hauptaufgaben und Komponenten des Systems und auch die Abläufe im System erkannt werden konnten. Das wichtigste Ergebnis der SGAM-Architekturphase (mit Component-, Information- und Communication Layer) war das Extrahieren der notwendigen Kommunikationswege zwischen den einzelnen Komponenten des Systems. Für diese Ergebnisse eignete sich das Vorgehen nach SGAM sehr gut. Andererseits waren einige Modellierungsschritte, wie zum Beispiel das Entwickeln von Sequenzdiagrammen für die einzelnen Primary Use Cases (Function Layer), das Zuordnen von Informationsobjekten und deren Datenmodellstandards zu *jedem* möglichen Kommunikationsweg (Information Layer) oder das Festlegen von verwendeten Protokollen bei der Kommunikation zwischen den Komponenten (Communication Layer) zu feingranular und zeitintensiv, da sie der Projektgruppe aus der jetzigen Sicht keinen nennenswerten Nutzen brachten. In der Rückschau erscheint daher das Vorgehen nach SGAM in Teilen sinnvoll, hätte aber in der Projektgruppe durchaus schlanker angewendet werden können.

Zuletzt wurde der Code an einigen Stellen unnötig kompliziert gestaltet. Mit dem Ziel, *Powder* möglichst erweiterbar zu programmieren, wurden viele abstrakte Klassen und Interfaces verwendet. Dadurch ist es im Nachhinein gut möglich, z. B. weitere Optimierungsalgorithmen in *Powder* zu integrieren. Der Nachteil ist allerdings, dass es ungemein schwerer wird, das Programmierte nachzuvollziehen. Dass dieser Umstand bereits innerhalb der Gruppe bemerkt wurde, spricht für sich.

Aus einigen negativen Aspekten konnten noch während der Projektdauer Handlungsempfehlungen gefolgert und auch umgesetzt werden. Es wurde zunächst festgestellt, dass die wöchentliche Neuzusammenstellung von Kleingruppen insofern nachteilig war, als sich teilweise keiner dafür verantwortlich fühlte, die Fertigstellung übriggebliebener Aufgaben voranzutreiben. Als Maßnahme wurden Modulverantwortliche benannt. Diese Personen waren im Rahmen ihres Moduls dafür verantwortlich, auf Vollständigkeit, Konsistenz, Qualitätssicherung und die Einhaltung des Zeitplans zu achten. Der Einführung der Modulverantwortlichen wurde auch im Nachhinein als gute Entscheidung angesehen.

Ein weiterer Aspekt ist der Umgang mit Planning-Poker. Diese Technik wurde zur zeitlichen Einschätzung von Aufgaben eingesetzt. Dabei wurde von einigen Projektmitgliedern kritisiert, dass Aufgaben keine sinnvollen Zeiteinschätzungen erhalten haben und die Verhältnisse zwischen manchen Aufgaben unrealistisch seien. In einer Retrospektive wurde allerdings festgestellt, dass die Ermittlung realistischer Zeitschätzung beim Planning-Poker nur zweitrangig ist. Einen viel größeren Nutzen bringt dagegen die Diskussion der einzelnen Aufgaben. Diese wurden bei der Zeitschätzung meist so weit erläutert, dass jeder eine Vorstellung davon hatte, was Bestandteil der jeweiligen Aufgabe war. Aus diesem Grund wurde das Planning-Poker bis zum Ende der Projektdauer beibehalten.

Zusammenfassend überwiegen die positiven Aspekte. Die negative Selbstkritik hat die Arbeitsmoral nicht gestört und die meisten Aspekte konnten noch während der Projektdauer genutzt werden, um das Vorgehen zu verbessern.

Rückblick auf die Umsetzung Im Rückblick auf das Ergebnis der Projektgruppe kann festgestellt werden, dass die Anforderungen zwar manchmal alternativ, jedoch den hintergründigen Zweck erfüllend, umgesetzt wurden. In wenigen Teilen wurden Anforderungen dagegen komplett gestrichen. Dies ist jedoch nur geschehen, wenn die ursprüngliche Anforderung im Laufe des Projekts nicht mehr benötigt wurde oder sich Rahmenbedingungen änderten. Zentrale funktionale Anforderungen an das System wurden entweder direkt oder im vorgesehenen Sinn implementiert. Dabei wurden verschiedene Optimierungsalgorithmen und Methoden des maschinellen Lernens auf die Problemstellung angewendet.

Das Kernmerkmal von *Powder* ist die Kombination von Einsatzplan- und Produktportfoliooptimierung, wohingegen herkömmliche Ansätze diese beiden Probleme getrennt betrachten. In *Powder* werden Marktrestriktionen berücksichtigt, realistische Anlagenmodelle verwendet und standardisierte Schnittstellen implementiert, sodass es in der Praxis verwendet werden kann. Die Praxistauglichkeit wird zusätzlich dadurch ermöglicht, dass die Optimierung zu einem wählbaren Zeitpunkt garantiert terminiert. Bereits bei sehr kurzen Optimierungszeiten von weniger als 15 Minuten werden sehr gute Ergebnisse erzielt. Außerdem ermöglicht *Powder* die automatisierte Berechnung der Angebotspreise der Produkte, abhängig von der vom VK-Betreiber gewünschten Annahmewahrscheinlichkeit. Insgesamt lässt sich sagen, dass das Ziel des Projekts, die Entwicklung eines Systems, welches das Agieren eines virtuellen Kraftwerks für den Day-Ahead-Handel am Energiemarkt gewinnmaximiert, erfolgreich umgesetzt wurde.

8.2. Ausblick

Der nächste Schritt bei der Weiterentwicklung von *Powder* ist die Anbindung an aktuelle Wetterdaten sowie die Benutzung realer statt simulierter Anlagen. Darüber hinaus gibt es einige Möglichkeiten, das System noch interessanter und besser zu machen.

Anlagensimulation Es könnten weitere Anlagentypen wie Windkraftwerke und Batterien modelliert werden. Die Integration entsprechender Anlagentypen könnte die Flexibilität der virtuellen Kraftwerke erhöhen und von spezifischen Wetterumständen unabhängig machen. Gerade die Einbindung von Batterien stellt in diesem Zusammenhang eine erhöhte Flexibilität dar und ist zudem eine Anforderung der Projektgruppe (L-03), die allerdings aus Komplexitätsgründen nicht bearbeitet wurde.

Markt- und Wetterprognose Die Marktprognose könnte die verschiedenen Prognose-Algorithmen kombinieren und neben der Temperatur auch von der Windstärke lernen. Sowohl Wetter- als auch Marktdaten könnten für weitere Jahre eingebunden werden. Damit würde es dem maschinellen Lernen möglich, Trends innerhalb von Jahren wie Jahreszeiten o. Ä. zu erkennen und dadurch die Marktprognose möglicherweise zu verbessern.

Produktportfoliooptimierung Wenn mehr Daten über das Zustandekommen des Marktpreises bekannt sind, kann der optimale Angebotspreis präziser berechnet werden. Das wäre eine weitere Optimierung. Am Folgetag wird bekannt, wie viel des Portfolios verkauft wurde. Die Optimierungsmodule könnten diese Daten mit in ihre Kalkulationen einbeziehen. Weitere Optimierungsverfahren (auch für die Einsatzplanoptimierung) könnten implementiert und dann mit den vorhandenen Möglichkeiten evaluiert werden.

Wenn *Powder* gute Ergebnisse liefert, dann kann es durch diese Weiterentwicklungen ein Bestandteil zur erfolgreichen und wirtschaftlicheren Integration von Anlagenpools von erneuerbaren Energien am Energiemarkt werden.

A. Daten für die Evaluation der Optimierer

In diesem Abschnitt sind für alle in der Evaluation der Optimierer durchgeführten Experimente die Wertebelegungen der Faktoren angegeben. Zusätzlich sind einige ausgewählte Abbildungen für den Evaluationsschritt 2c enthalten.

Experiment	Anlagenanzahl	VK-Leistung [kWh]	Anteil BHKW-Leistung	Fahrplananzahl
1	1319	4939,60	0,56	198
2	4637	4052,43	0,11	283
3	3205	2909,07	0,90	128
4	3763	3357,84	0,91	240
5	232	3985,98	0,01	110
6	4306	1886,28	0,22	30
7	2106	2393,54	0,79	228
8	2941	1221,47	0,40	40
9	927	36,80	0,70	80
10	1742	655,45	0,50	165

Tabelle 8: Daten für Evaluation der inneren Optimierer (s. Abschnitt 6.10.5) und für Evaluationsschritte 2b, 2c und 2d

Experiment	Anlagenanzahl	VK-Leistung [kWh]	Anteil BHKW-Leistung	Fahrplananzahl
1	33	2929,93	1,0	362
2	85	1900,60	1,0	445
3	48	1734,03	1,0	126
4	70	1094,72	1,0	203
5	10	663,50	1,0	67

Tabelle 9: Daten für Evaluationsschritte 1 und 2a

Experiment	Anlagenanzahl	VK-Leistung [kWh]	Anteil BHKW-Leistung	Fahrplananzahl
1	10	150,00	1,00	10
2	10	150,00	1,00	14
3	10	150,00	1,00	27
4	10	150,00	1,00	37
5	10	150,00	1,00	43
6	10	150,00	1,00	55
7	10	150,00	1,00	66
8	10	150,00	1,00	76
9	10	150,00	1,00	80
10	10	150,00	1,00	94

Tabelle 10: Daten für Evaluationsschritt 3a

Experiment	Anlagenanzahl	VK-Leistung [kWh]	Anteil BHKW-Leistung	Fahrplananzahl
1	10	158,31	1,0	10
2	10	305,08	1,0	10
3	10	475,80	1,0	10
4	10	604,33	1,0	10
5	10	733,09	1,0	10

Tabelle 11: Daten für Evaluationsschritt 3b, Design I

Experiment	Anlagenanzahl	VK-Leistung [kWh]	Anteil BHKW-Leistung	Fahrplananzahl
1	10	158,31	1,0	10
2	19	298,82	1,0	10
3	32	453,72	1,0	10
4	41	608,20	1,0	10
5	47	734,45	1,0	10

Tabelle 12: Daten für Evaluationsschritt 3b, Design II

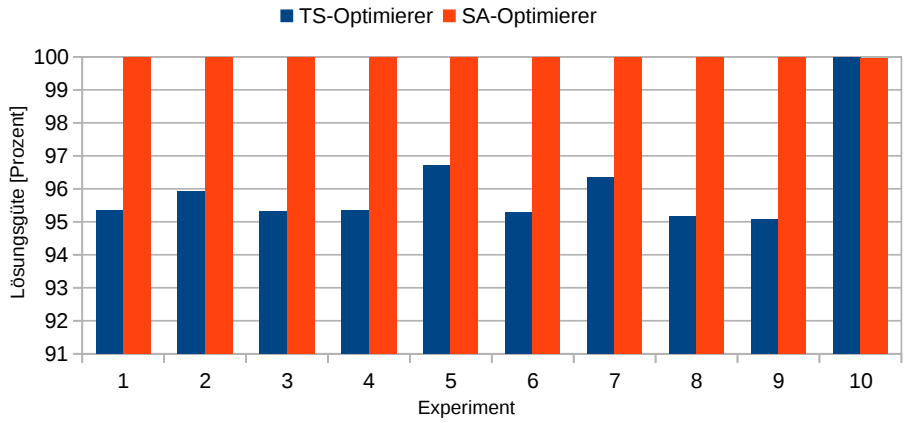


Abbildung 57: Lösungsgüte der äußeren Optimierer bei einer Laufzeit von 15 Minuten (Evaluationsschritt 2c)

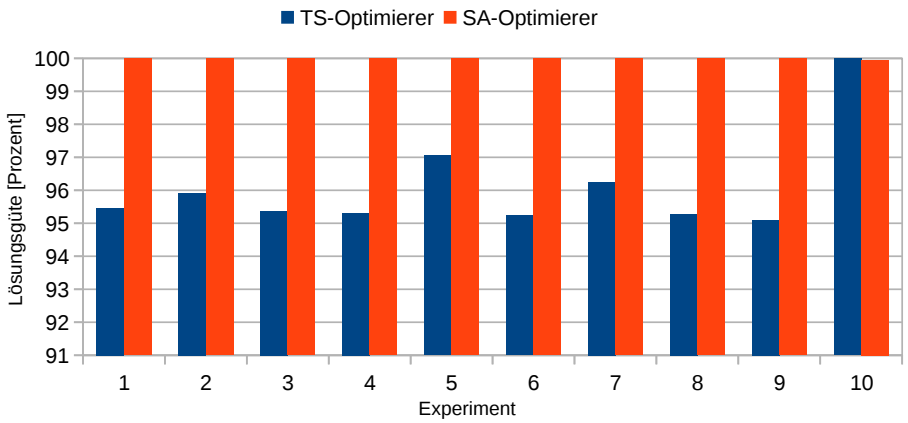


Abbildung 58: Lösungsgüte der äußeren Optimierer bei einer Laufzeit von 35 Minuten (Evaluationsschritt 2c)

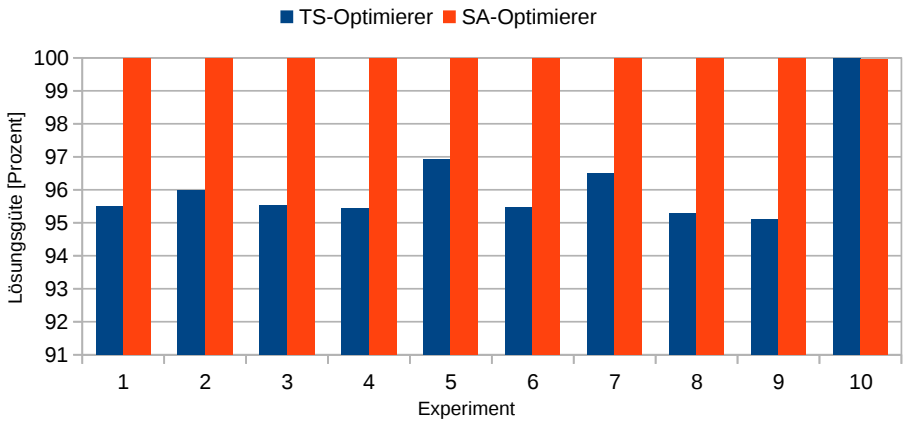


Abbildung 59: Lösungsgüte der äußeren Optimierer bei einer Laufzeit von 300 Minuten (Evaluationsschritt 2c)

B. Monatliche Sprintzusammenfassungen

Dieses Kapitel enthält in Monate aufgeteilte Zusammenfassungen aller Sprints. Die Zusammenfassung erfolgt chronologisch. Sie beschreibt in kurzen Worten durchgeführte Aktionen und bearbeitete Aufgaben. Während der Durchführung der Projektgruppe hat sich die Dauer der Sprints einmalig geändert. Der Übersichtlichkeit halber wurde die Zusammenfassung daher zusätzlich in Monate strukturiert. Die ersten sieben Sprints waren zweiwöchig, der achte Sprint dauerte wegen des Übergangs eineinhalb Wochen und ab dem neunten Sprint wurde die Dauer dann endgültig auf eine Woche festgelegt.

Mai

Im Mai begann die Projektgruppe den ersten Sprint, welcher vom 22.05.2015 bis zum 05.06.2015 andauerte.

Sprint 1 In diesem Sprint entschied sich die Projektgruppe für den Einsatz von JIRA/Confluence als Projektmanagement Software. Da die Lizenzen jedoch nicht sofort verfügbar waren, konnte in diesem Sprint noch kein JIRA verwendet werden. Außerdem wurde das Vorgehen für testgetriebene Software-Entwicklung erstellt und die Anforderungsanalysen mit Hilfe von SGAM auf Business-Layer-Ebene angefangen. Des Weiteren wurden Personen als Scrum-Master und Product Owner festgelegt.

Es wurde eine LaTeX-Vorlage für das Anforderungsdokument erstellt und ein Etherpad2LaTeX Python Skript für Protokolle erstellt. Für das Anforderungsdokument wurde die Vision zunächst als Grafik erstellt. Ebenso wichtiger Bestandteil dieses Sprints waren zwei Interviews mit den Betreuern (in ihrer Kundenrolle), wodurch das Projekt deutlicher definiert wurde.

Juni

Im Juni fand der letzte Teil des ersten Sprints und der zweite (05.06.2015 - 19.06.2015) sowie dritte (19.06.2015 - 03.07.2015) Sprint statt.

Sprint 2 Ab diesem Sprint konnten JIRA/Confluence eingesetzt werden. Die Vision wurde verschriftlicht und als Continuous Integration Software soll Bamboo eingesetzt werden.

Ein großer Bestandteil dieses Sprints waren Recherche-Aufgaben: Dazu zählen die Recherchen zu Evaluationsmethoden für den Optimierer, Recherchen zum Zugang zu Markt- und Wetterdaten, sowie Recherchen zu Simulationsumgebungen für die Anlagen.

Zentrales Element dieses Sprints war der Anforderungsworkshop mit den Betreuern. Dazu wurden drei Gruppen gebildet, in denen jeweils ein Betreuer war. In der ersten Runde musste sich jede Gruppe mit einem HLUC aus dem Business Layer beschäftigen und dazu PUC mittels Brainstorming herausfinden. Diese wurden auf blaue Moderationskarten geschrieben. Zusätzlich wurden die beteiligten Akteure auf grüne Moderationskarten geschrieben. Nach der Brainstorming-Phase wurden die Karten an der Tafel sortiert. Die zweite Runde mit den verbliebenen High Level Use Cases verlief analog.

Sprint 3 In diesem Sprint wurden Aktivitätsdiagramme zu den High Level Use Cases hinzugefügt. Darauf aufbauend konnten dann in Kleingruppen Aktivitäts- und Sequenzdiagramme zu den Primary Use Cases erstellt werden. Diese Aufgabe wurde so aufgeteilt, sodass die PUC zu einem HLUC von zwei Personen bearbeitet wurden. Aus den Primary Use Cases wurden außerdem Anforderungstabellen erstellt. Diese Anforderungstabellen sind angelehnt an die Use Case Tabellen des IEC/PAS 62559 Use Case Templates.

Neben der Arbeit mit SGAM und den Tabellen wurde die Struktur der Sitzungen verbessert. Für die internen Sitzungen und für Sitzungen mit Betreuern wurden Vorlagen im Etherpad erstellt und die Aufgaben des Moderators und des Protokollanten möglichst genau im Confluence definiert.

Juli

Im Juli fand der letzte Teil des dritten Sprints sowie der vierte (03.07.2015 - 17.07.2015) und fünfte (17.07.2015 - 31.07.2015) Sprint statt.

Sprint 4 Im vierten Sprint wurden die Information und Component Layer zu den einzelnen HLUC im SGAM erstellt. Durch die Bearbeitung durch Zweiergruppen entstanden Inkonsistenzen, die aufgelöst werden mussten. Ebenso zentral war die Recherche zu Kommunikationsmöglichkeiten innerhalb der Module des virtuellen Kraftwerks und mögliche Protokolle bzw. Standards zur Kommunikation mit der Systemumgebung. Unter anderem konnte bereits die Kommunikation zum Marktdatenprovider festgelegt werden.

Sprint 5 Im fünften Sprint wurde der Marktdaten-Scraper programmiert und ein Code-Git mit Maven-Projekt eingerichtet. Außerdem wurde der Communication Layer fertig gestellt und die möglichen, für das Projekt sinnvollen, Datenbanken recherchiert und oberflächlich evaluiert.

August

Teil des Augusts waren der sechste (31.07.2015 - 13.08.2015) und siebte (13.08.2015 - 27.08.2015) Sprint.

Sprint 6 In diesem Sprint wurde eine Planungsgruppe bestimmt, die zur Aufgabe hatte, das weitere Vorgehen der Projektgruppe bis zum Zwischenbericht zu planen. Zu diesem Zweck wurde ein Netzplan mit wichtigen Meilensteinen erstellt. Außerdem wurde eine theoretisches Modell für Blockheizkraftwerke recherchiert und schriftlich festgehalten. Weitere wichtige Aufgaben in diesem Sprint umfassten das Aufsetzen der Datenbank, Recherche für die Strukturierung des Zwischenberichtes und das Überführen der historischen Wetterdaten von der Epex-Spot-Webseite in eine Datei.

Sprint 7 Im siebten Sprint wurden viele einzelne Aufgaben erledigt. Dazu zählen die Erstellung eines Schaubilds der SGAM-Ebenen, das Aufsetzen eines Maven-Projekts, die Implementierung eines Parsers für die historischen Wetterdaten und die Herstellung der Konsistenz der SGAM-Diagramme.

Des Weiteren wurden verschiedene Technologien evaluiert, wie die mögliche Nutzung des Co-Simulationsframeworks Mosaik als Simulationsumgebung und des ORM-Frameworks Hibernate zur Abbildung von Java-Objekten auf relationale Datenbanken.

Eine Softwareentwicklungs-Meta-Gruppe hatte die Aufgabe, eine grobe Architektur für die Softwareentwicklung zu entwerfen. Zudem wurde ein theoretisches Modell für Photovoltaikanlagen erarbeitet. Einen wichtigen Meilenstein stellte die Entscheidung für ein Flexibilitätenmodell dar. Eine Erläuterung zu Flexibilitäten ist in Kapitel 6.8 nachzulesen.

September

Teil des Septembers war der Großteil des achten (27.08.2015 - 08.09.2015) sowie der neunte (08.09.2015 - 15.09.2015), zehnte (15.09.2015 - 22.09.2015) und elfte (22.09.2015 - 29.09.2015) Sprint.

Sprint 8 In Sprint 8 waren die beiden Hauptthemen der Zwischenbericht und der Beginn der Implementierung des Prototyps. Für den Zwischenbericht wurde zunächst eine Gliederung entwickelt und entsprechende LaTeX-Dokumente angelegt. Darauf wurden bereits einige zu schreibende Kapitel verteilt.

Zu Beginn der Implementierungsphase mussten noch grundlegende Entscheidungen getroffen werden. Daher wurden verschiedene Möglichkeiten für die interne Kommunikation der Module (u. a. RMI) evaluiert. Es erfolgte auch eine Gegenüberstellung verschiedener Logger-Frameworks. Des Weiteren wurde die Co-Simulationsumgebung Mosaik in das Projekt eingebunden und die historischen Markt- und Wetterdaten wurden in der Datenbank gespeichert. Zudem wurde damit begonnen, einfache Varianten des Flexibilitäten- und des Marktmoduls zu implementieren.

Sprint 9 In diesem Sprint wurde intensiv an der Zwischenberichtsdokumentation gearbeitet. Zu diesem Zweck wurden die verschiedenen Unterkapitel verteilt und von den jeweiligen Gruppenmitgliedern verfasst. Dazu zählten überwiegend Themen aus dem Bereich Projektmanagement, Entwurf und Technologien.

Auch bei der Implementierung des Prototyps konnten viele Fortschritte erzielt werden. Die einfachen Versionen des Flexibilitäten- und des Wettermoduls wurden fertig gestellt. Zudem wurde am einfachen Markt- und Optimierungsmodul gearbeitet. Zusätzlich wurden verschiedene Testframeworks (JUnit und TestNG) evaluiert und infolgedessen eine Entscheidung für TestNG getroffen.

Außerdem wurde die interne Kommunikation zwischen den Modulen über RMI realisiert und der eigene Logger implementiert, der den Logger Log4j erweitert.

Sprint 10 Auch im zehnten Sprint wurde intensiv an der Zwischenberichtsdokumentation gearbeitet. Der Schwerpunkt lag hier in den Kapiteln Projektmanagement, Anforderungen, Entwurf und Implementierung.

Für die Umsetzung des Prototyps wurde ein einfacher Optimierer implementiert, die Wetterdaten wurden in der Datenbank gespeichert und die Marktdatenprognose enthält Stundenpreise. Des Weiteren entstanden erste Ausgaben für eine mögliche Live-Demo.

Außerdem wurde die Struktur und eine LaTeX-Vorlage für die Zwischenpräsentation erstellt.

Sprint 11 Dieser elfte Sprint war der letzte komplette Sprint vor der Abgabe des Zwischenberichts. Dementsprechend stand die Vervollständigung des Zwischenberichtdokuments im Fokus. Neben vereinzelten noch fehlenden (Unter-) Kapiteln wurde ein Zwischenfazit ergänzt, in dem der bisherige Projektablauf reflektiert und das weitere anschließende Vorgehen beschrieben wurde.

Außerdem wurde der gesamte Zwischenbericht von den Lektoren und weiteren Freiwilligen, die noch Kapazitäten besaßen, durchgelesen und (sofern notwendig) korrigiert.

Oktober

Im Oktober wurde der zwölfte (29.09.2015 - 06.10.2015) noch im September begonnene Sprint beendet. Sprint 13 (06.10.2015 - 13.10.2015) war der letzte noch auf die alte Art benannte Sprint. Ab dem nächsten wurden die Sprints anhand der Kalenderwoche identifiziert, in der sie durchgeführt wurden. Daher sind Sprint KW42 (13.10.2015 - 20.10.2015) und Sprint KW43 (20.10.2015 - 27.10.2015) die neuen Namen für die in diesem Monat noch komplett durchgeführten Sprints.

Sprint 12 Der zwölfte Sprint endete einen Tag vor Abgabe des Zwischenberichtes. Der Großteil der Aufgaben bestand aus dem Beginn der Arbeit an der Zwischenpräsentation. Zusätzlich wurde ein Brainstorming zur Umsetzung des Optimierers durchgeführt und das PV-Modell der Simulation ausgehend vom evolutionären Prototyp erweitert.

Sprint 13 Während dieses Sprints wurde die Zwischenpräsentation vollständig vorbereitet und fand sehr erfolgreich statt. Dazu wurde unter anderem eine Skype-Konferenz mit Prof. Lehnhoff eingerichtet, der teilnehmen wollte, sich jedoch außerorts befand. Weiterhin wurde ein Netzplan erstellt, der das Vorgehen bis Weihnachten genauer darstellte.

Im Flexibilitätenmodul wurden Kraftwerksstammdaten eingeführt und angelegt, welche einzelne Arten von Anlagen definieren. Zu den Anlagentypen BHKW und PV wurden Kostenfunktionen recherchiert. Für die Marktprognose wurde Machine Learning aus den Merkmalen Wetter und Datum implementiert.

Sprint KW42 Zentraler Teil dieses Sprints war die Durchführung eines Workshops zur Auswahl geeigneter Optimierungsverfahren. An diesem haben auch die Betreuer teilgenommen. Die einzelnen betrachteten Verfahren wurden auf kleinere Gruppen aufgeteilt und innerhalb der Gruppen wurde überlegt, wie mit Hilfe der Verfahren das Optimierungsproblem gelöst werden kann.

Abgesehen vom Workshop wurde die Eingabe eines zentral gesetzten Startdatums implementiert. Die Berechnung der Flexibilität einzelner Anlagen wurde verfeinert. Unter anderem wurde hier für die PV-Anlagen die Sonneneinstrahlung von horizontaler in vertikale Einstrahlung umgerechnet. Für die Marktprognose wurde herausgefunden, welche Einwirkung unterschiedliche Gewichtungsmethoden für die ähnlichen Tage bei der Erstellung der Marktpreise haben.

Sprint KW43 In diesem Sprint wurde dem Autor der SGAM Toolbox (Einer Erweiterung des Enterprise Architect) Rückmeldung zur Nutzung derselben zugesandt.

Im Flexibilitätenmodul wurden verschiedene Fehler behoben und deren erneutes Auftreten durch das Hinzufügen entsprechender Tests abgedeckt. Zusätzlich wurde die Darstellung der Zustände der BHKW-Anlagen betrachtet und überlegt wie diese umgesetzt werden soll. Die Marktprognose wurde mit dem Framework Weka über Machine Learning implementiert. Zusätzlich wurden Einheitenklassen eingeführt, damit immer klar ist, mit welchem Wert in welcher Einheit gerechnet wird.

November

Im November sind fünf Sprints (unter Einbeziehung des Sprints in der 44. Kalenderwoche, welche auch im Oktober lag) durchgeführt worden: Sprint KW44 (27.10.2015 - 03.11.2015), Sprint KW45 (03.11.2015 - 10.11.2015), Sprint KW46 (10.11.2015 - 17.11.2015), Sprint KW47 (17.11.2015 - 24.11.2015) und Sprint KW48 (24.11.2015 - 01.12.2015).

Sprint KW44 Während dieses Sprints wurden die vom Deutschen Wetterdienst und der universitätsinternen Wetterstation erhaltenen, unverrauschten Wetterdaten in die Datenbank übertragen.

Es wurde während des Optimierungsworkshops angedacht, dass das Rucksackproblem ein möglicher Lösungsweg sein könnte. Daher wurde in diesem Sprint versucht, das Rucksackproblem auf das Optimierungsproblem abzubilden. Da die Probleme jedoch nur verwandt, jedoch nicht aufeinander reduzierbar sind, konnte dies nicht geschehen. Die Beschäftigung hat jedoch bei der Aufstellung des Optimierungsproblems als Formel geholfen.

Zusätzlich wurden erneut, diesmal jedoch andere, Fehler im Flexibilitätenmodul behoben. Ebenfalls wurde mit der Implementierung von Einschränkungen für die Flexibilität der BHKWs begonnen. Diese legen beispielsweise fest, wie lange ein BHKW am Stück abgeschaltet werden sein muss, bevor es wieder hochgefahren werden kann. Es wurde zudem begonnen, Simulated Annealing als Lösung für die Einsatzplanoptimierung zu implementieren.

Sprint KW45 Es wurde ein gruppenweites Codereview über die bisherigen Teile des Projektes durchgeführt. Dabei wurden verschiedene kleinere Änderungen angedacht, die die Qualität des Codes verbessern sollten. Diese wurden teilweise schon während dieser Woche umgesetzt. Zusätzlich wurden in diesem Sprint viele der in den vorherigen zwei Sprints begonnenen Aufgaben abgeschlossen.

Die Einschränkungen für die BHKWs wurden noch erweitert und fertig implementiert. Das Modell der PV-Anlagen wurde mit Stammdaten erweitert. Die Einheitenklassenanpassung im gesamten Projekt ist fertiggestellt worden.

Sprint KW46 Für Bamboo wurden E-Mail-Benachrichtigungen aktiviert, um immer schnell auf Fehler aufmerksam gemacht zu werden.

Aus dem Codereview der vorherigen Woche wurde die Aufgabe aufgenommen, die Zufallswerte innerhalb des Projektes deterministisch zu erzeugen. Es wurde das Auslesen von Stammdatendateien und anderen Hilfsdateien implementiert. Zur Lösung der Produktportfoliooptimierung wurde ein Optimierer entwickelt, welcher rekursiv möglichst große Intervalle mit möglichst viel Leistung besetzt. Zusätzlich wurde alternativ dazu über Lineare Programmierung recherchiert und ein Tool gesucht, mit welchem dieser Ansatz implementiert werden konnte. Die Implementierung des Simulated Annealing

Optimierer-Prototyps wurde abgeschlossen. Im Flexibilitätenmodul wurde für die BHKWs ein Anlagenpool erzeugt, welcher diese verwaltet. Da für die Marktprognose weitere Daten benötigt wurden, wurde der Scaper um die Auslesung dieser erweitert. Zusätzlich wurde begonnen, die Parameter für das Machine Learning der Marktprognose zu verfeinern.

Sprint KW47 Dieser Sprint stellt den Zeitpunkt dar, an dem begonnen wurde, die externe Kommunikation mittels des Industriestandards OPC UA zu realisieren. Zunächst wurde hierzu recherchiert, mit welchen Tools dies geschehen könnte. Bei der Recherche über Lineare Programmierung ist aufgefallen, dass diese möglicherweise genutzt werden kann, um das Gesamtproblem zu lösen. Daher wurde diese Möglichkeit nochmal untersucht. Gurobi (s. Abschnitt 5.14) wurde als Tool gewählt, um Lineare Programmierung als Löser zu verwenden. Zur weiteren Nutzung wurde aus dem Simulated Annealing Optimierer, die allgemeine Struktur eines Evolutionären Algorithmus herausgelöst und als abstrakte Klasse weitergenutzt. Es wurde damit begonnen, eine Tabusuche als Lösung für die Einsatzplanoptinierung zu implementieren. Das Flexibilitätenmodul wurde angepasst, sodass die Anzahl verschiedener parametrisierter Anlagentypen eingestellt werden kann. Es wurde recherchiert, welche Rahmenbedingungen für die Vermarktung vorliegen.

Sprint KW48 Für OPC UA wurde die Spezifikation analysiert. Die Evaluation des Gesamtsystems wurde grob geplant. Zur Nutzung des Systems wurde es ermöglicht, einen beispielhaften Optimierungslauf zu starten und Einstellungen an alle Module zu vermitteln. Weitere historische Wetterdaten wurden in die Datenbank übernommen. Für die Flexibilitätenberechnung der BHKWs wurde der Einfluss des Wärmeverbrauchs verschiedener Verbraucherprofile implementiert.

Dezember

Im Dezember wurden die Sprints KW49 (01.12.2015 - 08.12.2015), KW50 (08.12.2015 - 15.12.2015) und KW51 (15.12.2015 - 22.12.2015) durchgeführt. Zwischen dem 23.12.2015 und dem 05.01.2016 hat die Projektgruppe Weihnachtsferien gemacht.

Sprint KW49 In diesem Sprint wurde die genauere Planung für das kommende Jahr 2016 erstellt.

Es wurde ein größerer Test implementiert, welcher überprüft, ob die Ausführung des Projektes deterministisch zum selben Ergebnis kommt. Weitere parametrisierte BHKW-Anlagentypen wurden implementiert und das Mapping von BHKWs auf Haushaltsprofile angepasst. Es wurde die parametrisierte Erstellung von Optimierern mithilfe des `BuilderPatterns` entworfen. Die Marktprognose wurde verbessert, indem weitere Attribute (wie z. B. der Gaspreis) vom Machine Learning gelernt werden.

Sprint KW50 In diesem Sprint gab es noch einmal ein Codereview, welches wiederum dazu diente, die Codequalität zu prüfen und das Wissen aller Mitglieder umfassend zu erweitern. Der im vorherigen Sprint erstellte Projektplan wurde noch einmal überarbeitet.

Der Entwurf für den `OptimizerBuilder` wurde implementiert. Die Modellierung des Wärmespeichers für BHKWs wurde verbessert, indem auch der Wärmeverlust über die Zeit betrachtet wurde. Es

wurde zusätzlich zum Auslesen auch das Speichern von Dateien implementiert. Die Arbeit an einem GUI Prototypen wurde innerhalb dieses Sprints begonnen.

Sprint KW51 Während dieses Sprints wurde das Evaluationsmodul erstellt und die konzeptionelle Ausführung desselben umgesetzt. Für das Flexibilitätenmodul wurde implementiert, dass weitere parametrisierte Anlagentypen hinzugefügt werden könnten. Die Marktprognose wurde metaoptimiert. Es wurde begonnen, das Projekt auf den Server auszulagern. Der GUI Prototyp wurde zum Click Prototyp erweitert.

Januar

Im Januar wurden die Sprints KW1 (06.01.2016 - 12.01.2016), KW2 (12.01.2016 - 19.01.2016), KW3 (19.01.2016 - 26.01.2016) und KW4 (26.01.2016 - 02.02.2016) durchgeführt.

Sprint KW1 In diesem Sprint wurde begonnen, den Dekodierer von Jörg Bremer in das Projekt einzubinden. Dazu wurde zunächst das Suchraummodell in die Problemdefinition eingebaut. Es wurde an der Umsetzung eines OPC UA-Clients gearbeitet, wozu zunächst mit der Erstellung eines Nodesets in Form einer XML-Datei begonnen wurde. Weiterhin wurde an der Auslagerung des Systems auf den Server gearbeitet, wo verschiedene Probleme aufgetreten sind. Die Metaoptimierung der Parameter der implementierten Optimierungsverfahren wurde begonnen.

Außerdem wurde in diesem Sprint mit der Vorbereitung der Präsentation der Projektgruppe auf dem Schülerinformationstag begonnen.

Sprint KW2 In diesem Sprint wurde die Auslagerung aller Module auf den Server erfolgreich abgeschlossen. Einige kleinere Fehler im Flexibilitätenmodul wurden behoben und dieses abschließend fertiggestellt. In den Optimierungen wurde das Nutzen des Dekodierers von Jörg Bremer ermöglicht. Für die Evaluation wurde die Auswertung der Evaluationsdurchläufe implementiert. Für eben diese wurde die Generierung einer Zusammenstellung des virtuellen Kraftwerks zum entsprechenden Szenario umgesetzt. Erzeugte Evaluationsszenarien sollten konsistent abgespeichert werden, daher wurde dafür begonnen, ein Stück Code zu entwickeln, welches sie abspeichern sollte. Die GUI wurde um die Anzeige von Optimierungsergebnissen in Diagrammform erweitert.

Der Vortrag auf dem Schülerinformationstag wurde weiter vorbereitet.

Sprint KW3 In diesem Sprint fand die Präsentation auf dem Schülerinformationstag erfolgreich statt. Es wurde weiter an der Marktprognose gearbeitet, welche mittlerweile auch die Wetterprognose mit übernommen hat. Dabei sind Fehler aufgefallen, welche in bei der Verbindung des Vorhersageframeworks Weka mit dem Testramework TestNG aufgetreten sind. Diese wurden ebenfalls aufgehoben. Ebenfalls wurde die Durchführung der Szenarien in der Evaluation weiter vorangetrieben. Für OPC UA wurde begonnen, Instanzen aus der fertigen XML-Datei mit dem Modell zu erstellen.

Sprint KW4 Während dieses Sprints war ein Großteil der Projektmitglieder im Urlaub und es wurden infolgedessen weniger Aufgaben bearbeitet, als in anderen Sprints.

Es wurde die Infrastruktur für die Evaluation der inneren Optimierer angelegt, sodass diese ausgeführt werden könnte. Zusätzlich wurde die Metaoptimierung der Optimierer abgeschlossen. Zudem wurde die Arbeit am Programmteil, welcher die Evaluationsszenarien abspeichert, begonnen.

Februar

Im Februar wurden die Sprints KW5 (02.02.2016 - 09.02.2016), KW6 (09.02.2016 - 16.02.2016), KW7 (16.02.2016 - 23.02.2016) und KW8 (23.02.2016 - 01.03.2016) durchgeführt.

Sprint KW5 Die Arbeit am Programmteil welcher die Evaluationsszenarien abspeichert wurde abgeschlossen. Für die Evaluation wurde es ermöglicht verschiedene Ziele unabhängig voneinander auszuführen. Zur Speicherung der jeweiligen Ergebnisse wurde es ermöglicht, dass diese anschließend in eine CSV-Datei gesichert werden. Aus dem Modell für OPC UA wurden Java-Klassen generiert, welche in das Projekt eingebunden werden können.

Sprint KW6 In diesem Sprint wurde die Ausführung der Evaluation weiter vorbereitet. Dazu wurde die Speicherung der Fitnesswerte aller Optimierer während der Optimierung ermöglicht, um einen Verlauf erhalten und bewerten zu können. Es wurde jetzt möglich gemacht, den kombinierten Optimierer zu nutzen, um eine optimale Lösung zur Nutzung als Vergleichspunkt zu berechnen. Ebenfalls wurde die Zählung der Fitnessfunktionsaufrufe aller Optimierer implementiert. Ein Fehler, welcher die Ausführung der Evaluation behinderte, wurde ebenfalls gefixt. Die Optimierungsheuristiken wurden erweitert, sodass sie jetzt auch berücksichtigen, wie viele Iterationen schon vergangen sind. Die Anlagendaten, welche bisher nur im Flexibilitätenmodul vorlagen, wurden in das VPP-Modul verschoben und werden bei Bedarf vom Flexibilitätenmodul angefordert. Mit dem Ziel, eine genauere Darstellung zu implementieren, wurde genauer recherchiert, welche Zuschläge und Kraftstoffkosten bei den einzelnen BHKWs anfallen. Für die Kommunikation mit OPC UA wurden OPC UA-Clients aufgesetzt, welche mit dem Server in Verbindung treten können. Die Marktprognose wird jetzt gecacht, um unnötige Ladezeiten nach dem ersten Zugriff zu vermeiden. Der Endbericht wurde neu strukturiert, um eine bessere Übersichtlichkeit zu garantieren.

Sprint KW7 Die Evaluation wurde um die Möglichkeit erweitert, sowohl nur die inneren Optimierer, als auch äußere Optimierer auszuführen und die Ergebnisse zu loggen. Fahrpläne der PV-Anlagen wurden von der Mutierung ausgeschlossen, da sie nur eine einzige sinnvolle Flexibilität besitzen. Die Kommunikation mit OPC UA wurde weiter vorangebracht. Jetzt können die übertragenen, nochmals aktualisierten CIM-Objekte (mit Ausnahme der Fahrpläne) korrekt in das Projekt übernommen werden. Ebenfalls können die Anlagendaten von den Anlagen aus an den Server gesendet und dort korrekt eingebunden werden. Zusätzlich wurde die Verbindung verschlüsselt. Die Marktprämien für BHKW- und PV-Anlagen wurden in die Gewinnberechnung eingebunden. Zusätzlich wurde die Anlagensimulation der PV-Anlagen um Modultypen erweitert und die Berechnung entsprechend verbessert. Die Auslagerung auf den Server wurde von letzten Fehlern befreit und erlaubt die Ausführung der Evaluation und des Projektes selbst auf dem Server. Die Marktprognose wurde ein weiteres Mal begonnen,

zu verbessern, indem die Parameter der Supportvektor Regression optimiert wurden. Es wurde begonnen, das Paper für den Environmental Informatics Price for Students zu schreiben. Die Bearbeitung und Verbesserung des Papers zieht sich bis zu dessen Abgabe nach Abschluss der Projektlaufzeit durch die Sprints und wird nicht erneut erwähnt. Die erwähnte neue Struktur des Endberichts wurde in diesem Sprint umgesetzt.

Sprint KW8 Die Optimierung der Marktprognose wurde abgeschlossen. Letzte Fehler im Flexibilitätenmodul wurden behoben. Die Kommunikation mit OPC UA wurde abgeschlossen, ðdie Übertragung der Fahrpläne an die Anlagen wurde ermöglicht. Es wurde nach der Umstellung des Endberichts erstmals umfassend mit der Überarbeitung der bisherigen Teile und dem Schreiben der neuen Teile begonnen. In den vorherigen Wochen sind schon einige kleinere Teile geschrieben worden, welche ebenfalls noch eingebunden werden müssen.

März

Im März fanden die letzten Sprints der Projektgruppe statt, also Sprint KW9 (01.03.2016 - 08.03.2016), KW10 (08.03.2016 - 15.03.2016), KW11 (15.03.2016 - 22.03.2016), KW12 (22.03.2016 - 29.03.2016) und der letzte Sprint KW13 (29.03.2016 - 31.03.2016), welcher keine komplette Woche mehr umfasste.

Sprint KW9 In diesem Sprint wurden einige Fehler in den Optimierern beim Ausführen der Evaluation entdeckt, welche erst aus den Ergebnissen der Evaluation ersichtlich wurden. Während dieses Sprints wurde begonnen, diese Fehler zu beheben. Für die Evaluation wurde die Generierung der Szenarien und ihrer `ParameterSets` effizienter gemacht, damit diese bei der Generierung größerer Mengen nicht zum Absturz des Projektes führen. Die Durchführung der Evaluation konnte durch die Fehler zwar begonnen, jedoch nicht abgeschlossen werden. Bei der Dokumentation sind weitere Kapitel vorläufig abgeschlossen worden.

Sprint KW10 In diesem Sprint wurden die zwei genutzten Dekodierer gegeneinander evaluiert und einer der beiden für den weiteren Verlauf der Evaluation ausgewählt. Nach Absprache mit den Kunden wurde klar, das noch ein weiteres Feature umgesetzt werden musste. Dieses beinhaltet die Implementierung einer Sicherheit bei der Wahl des Marktpreises, sodass das angebotene Produkt möglichst verkauft wird. Dies wurde umgesetzt. Eine kurze Recherche offenbarte ebenfalls, dass die Wahl der Produkte am Markt in 0,1 MW Intervallen geschehen muss. Diese Einschränkung wurde ebenfalls in das Projekt mit aufgenommen und umgesetzt. Weitere Arbeit an der Dokumentation wurde durchgeführt.

Sprint KW11 In diesem Sprint wurde begonnen, die Auslieferung des Projektes vorzubereiten. Nachdem kurz vor Beginn des Sprints der letzte der in Sprint KW9 entdeckten Fehler im kombinierten Optimierer gefunden und behoben wurde, konnte jetzt ebenfalls die Evaluation beginnen. Evaluationsszenarien wurden für den noch vorhandenen Zeitrahmen gewählt und in diesem Sprint durchgeführt. Dazu wurde ebenfalls der zum Zwischenbericht entwickelte Prototyp eines Optimierers

eingebunden und verglichen. Die Ergebnisse derselben standen erst nach dieser Woche komplett zur Verfügung, weshalb die Auswertung noch nicht durchgeführt werden konnte. Die GUI wurde zusätzlich zu den geplanten Aufgaben noch erweitert und auf den aktuellen Stand des Backends gebracht. Weitere Arbeit an der Dokumentation wurde durchgeführt, sodass die restliche Arbeit komplett ersichtlich wurde und für den letzten Sprint geplant werden konnte.

Sprint KW12 In diesem Sprint wurden alle übrigen Dokumentationsaufgaben zugewiesen und für den Abschluss vorbereitet. Die Evaluationsergebnisse wurden ausgewertet und ebenfalls dokumentiert. Der Code wurde angesehen und um letzte Kleinigkeiten, Anmerkungen und Hinweise für mögliche Weiterarbeit erweitert.

Sprint KW13 In diesem Sprint wurden letzte bzw. weitere Überarbeitungen durch die Lektoren in der Dokumentation und an dem Paper vorgenommen. Zusätzlich wurde die Abschlusspräsentation weiter vorbereitet.

C. Abbildungen

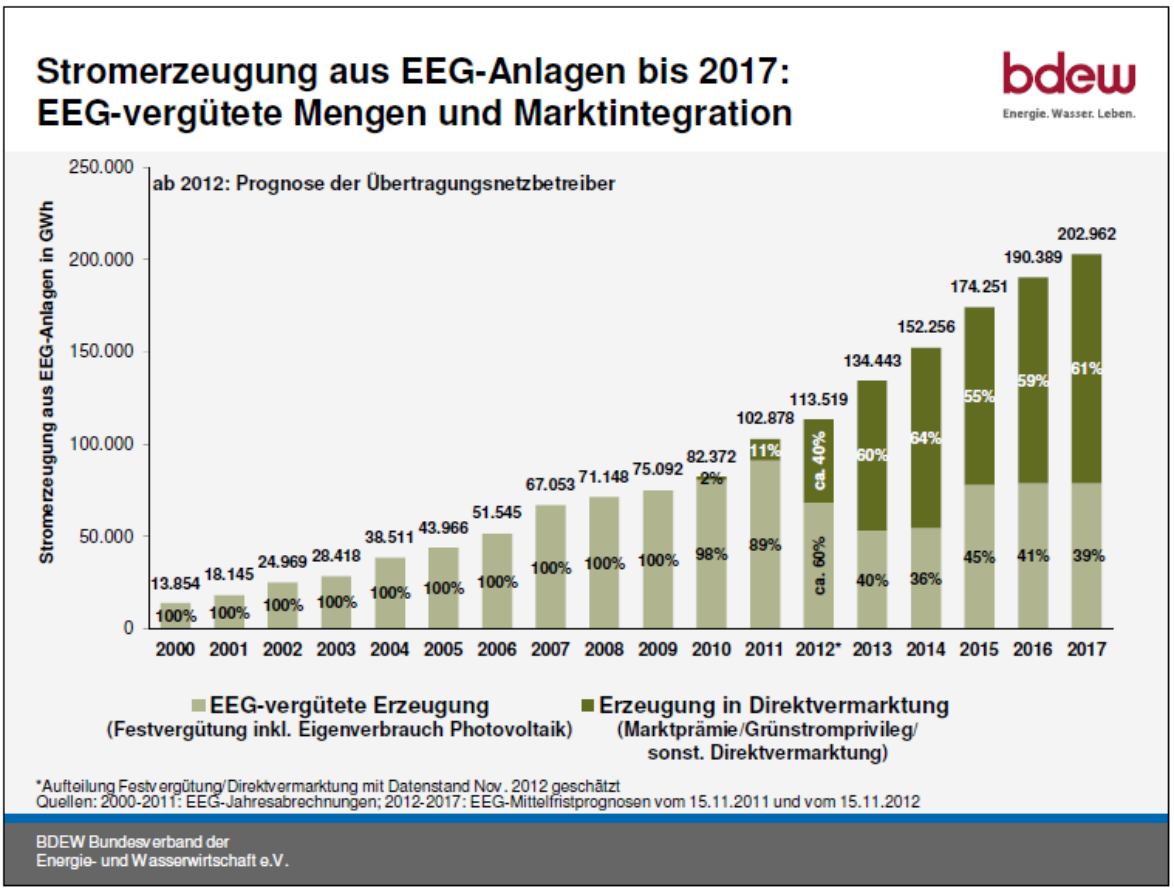


Abbildung 60: Stromerzeugung aus EEG-Anlagen bis 2017

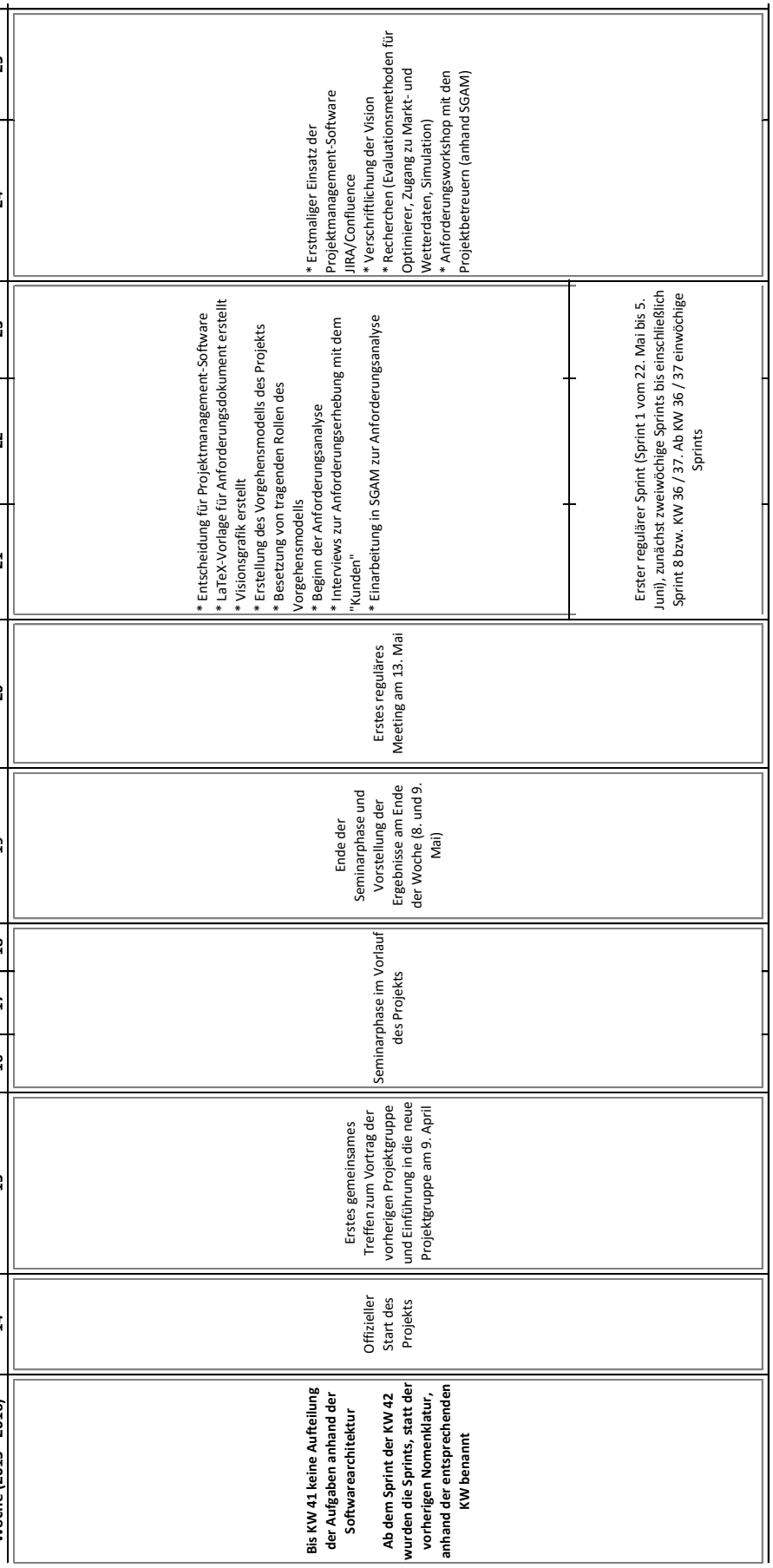


Abbildung 61: Netzplan des Projekts (1/6)

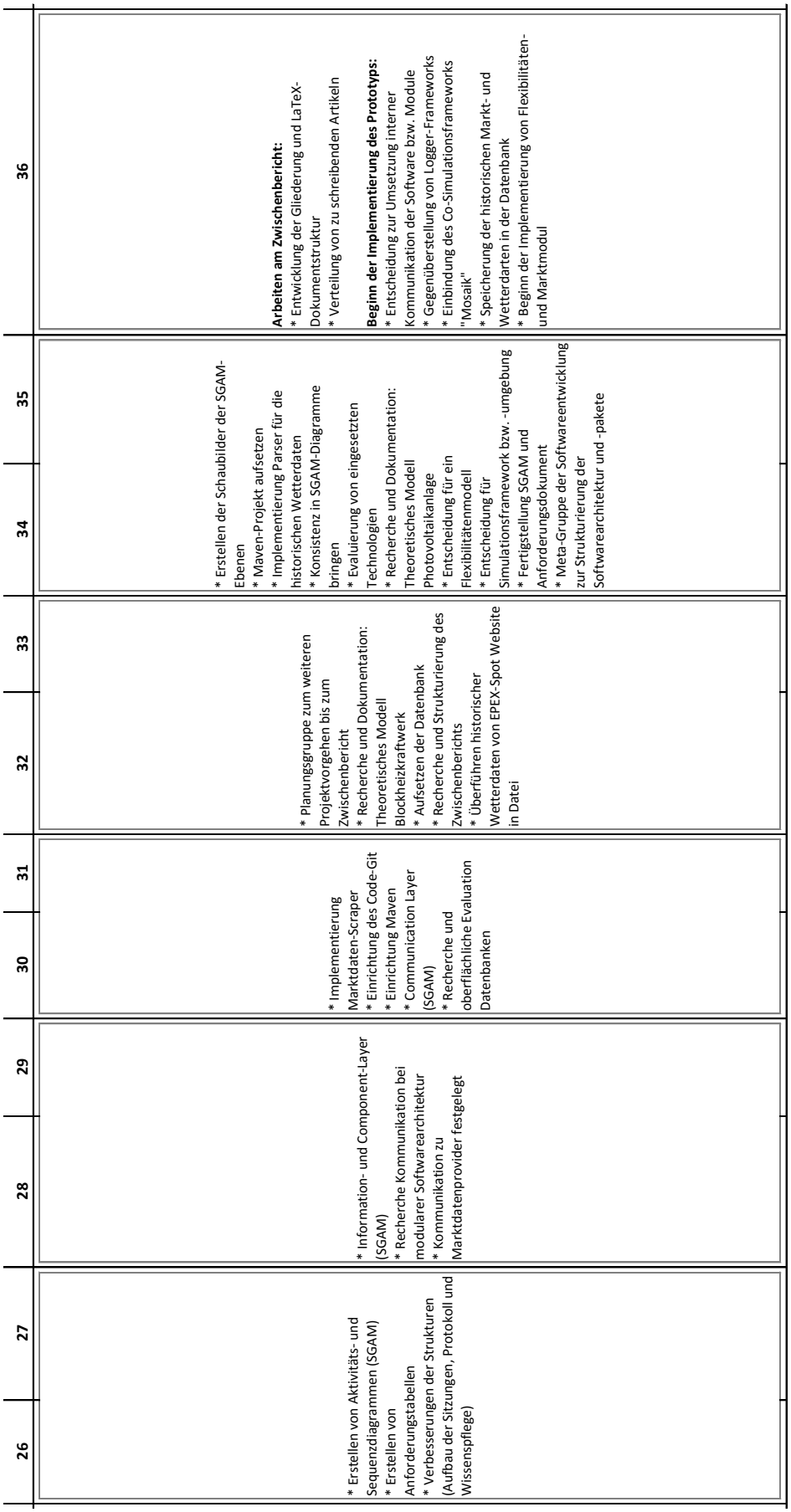


Abbildung 62: Netzplan des Projekts (2/6)

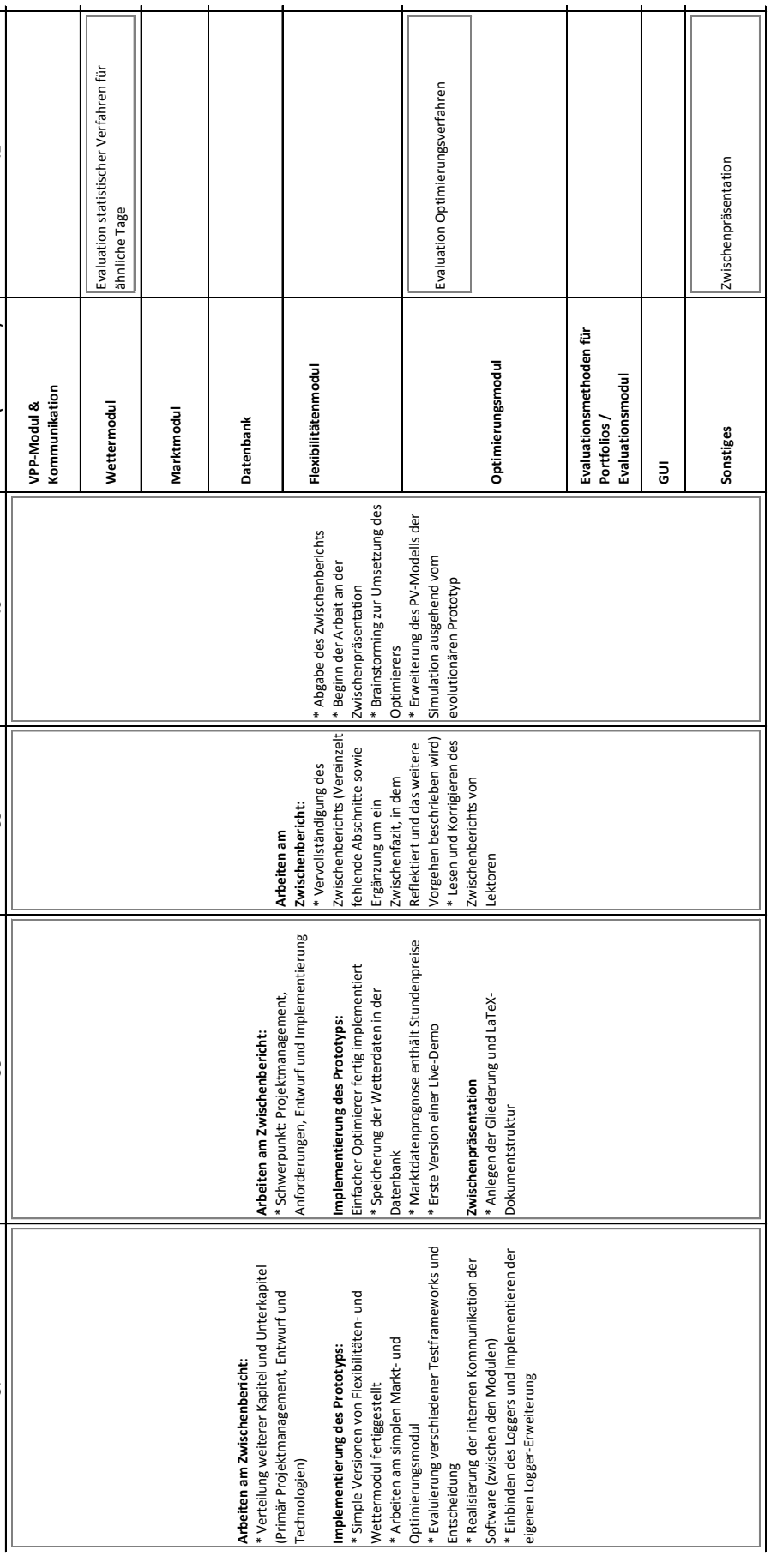


Abbildung 63: Netzplan des Projekts (3/6)

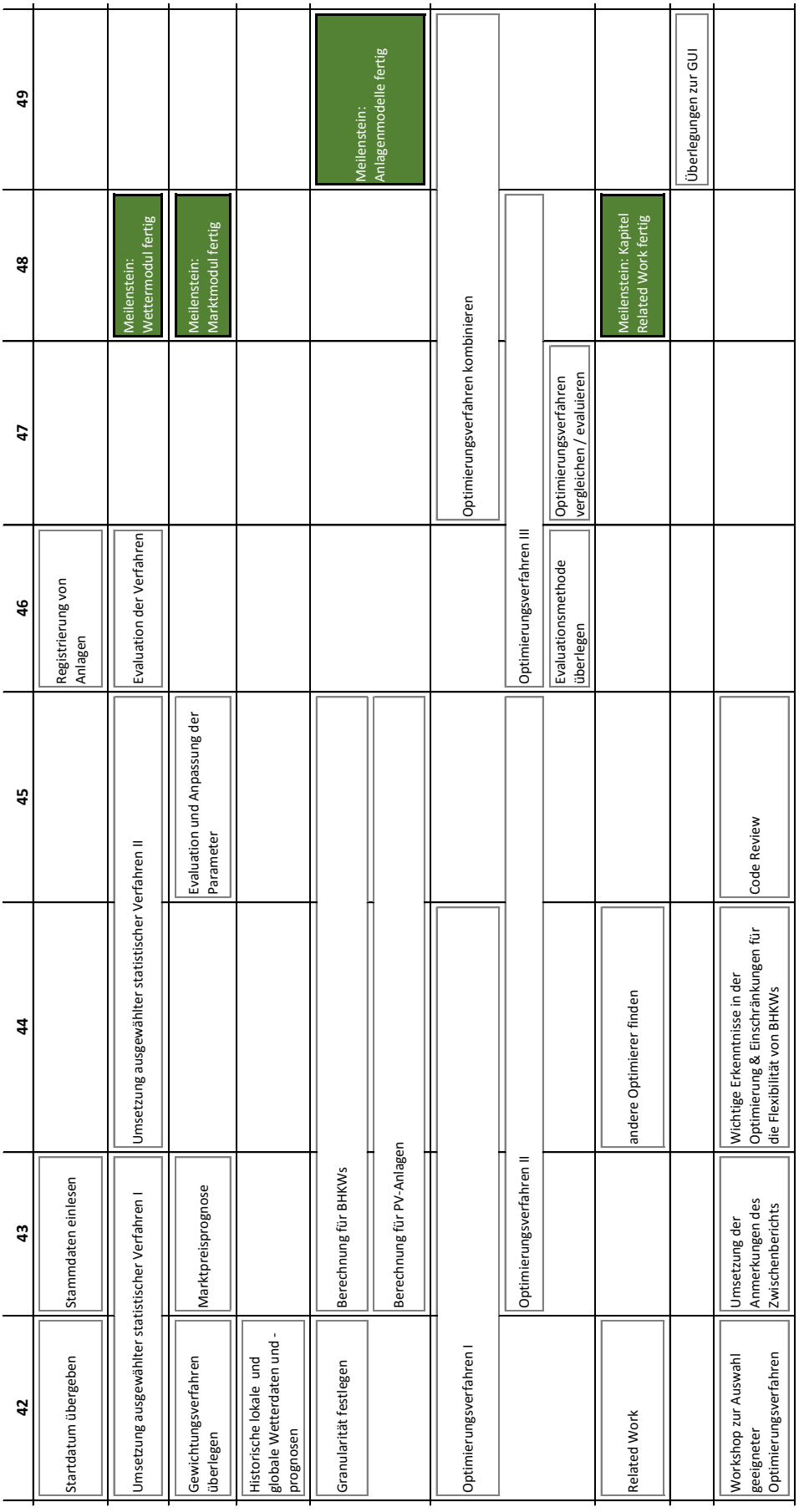


Abbildung 64: Netzplan des Projekts (4/6)

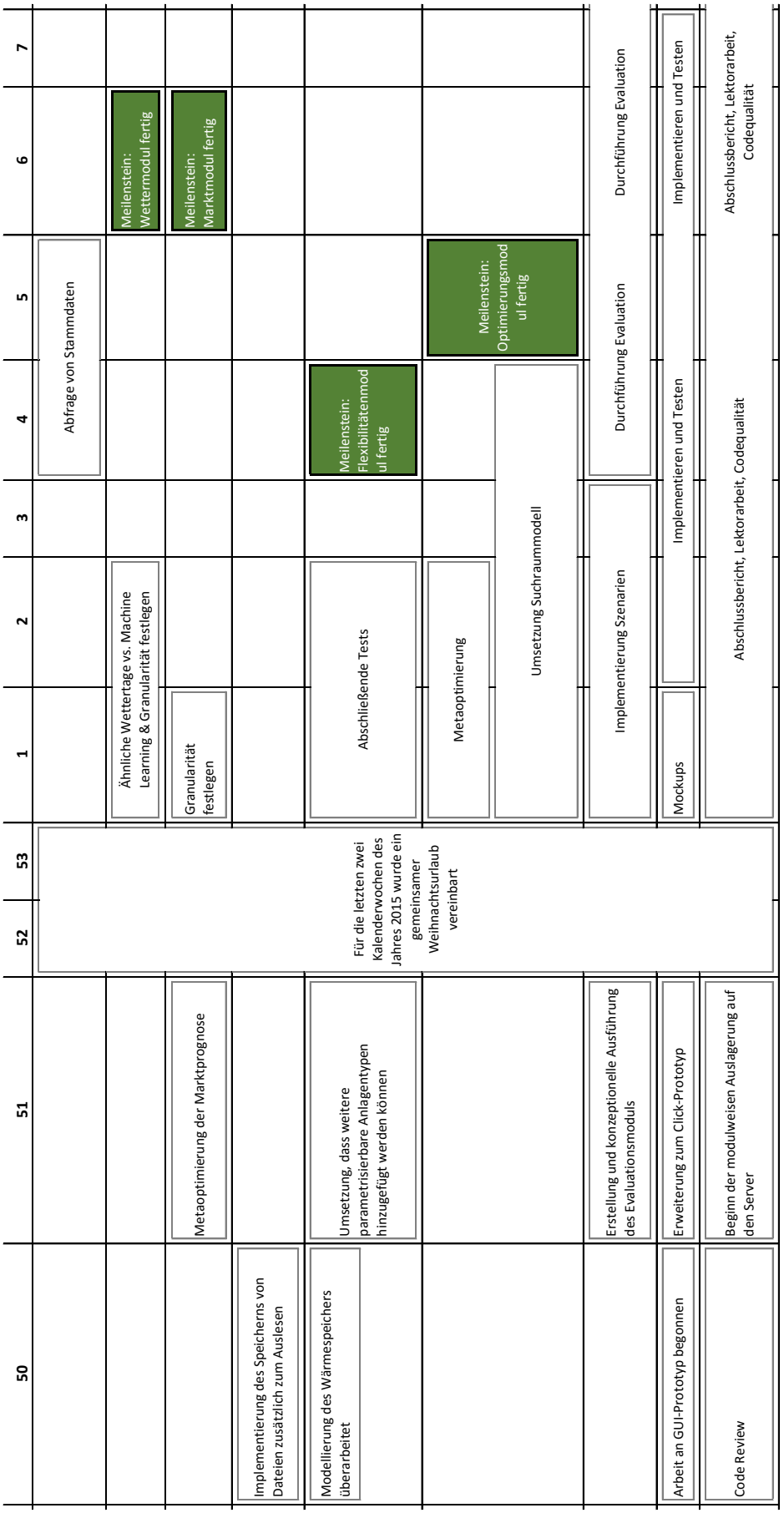


Abbildung 65: Netzplan des Projekts (5/6)

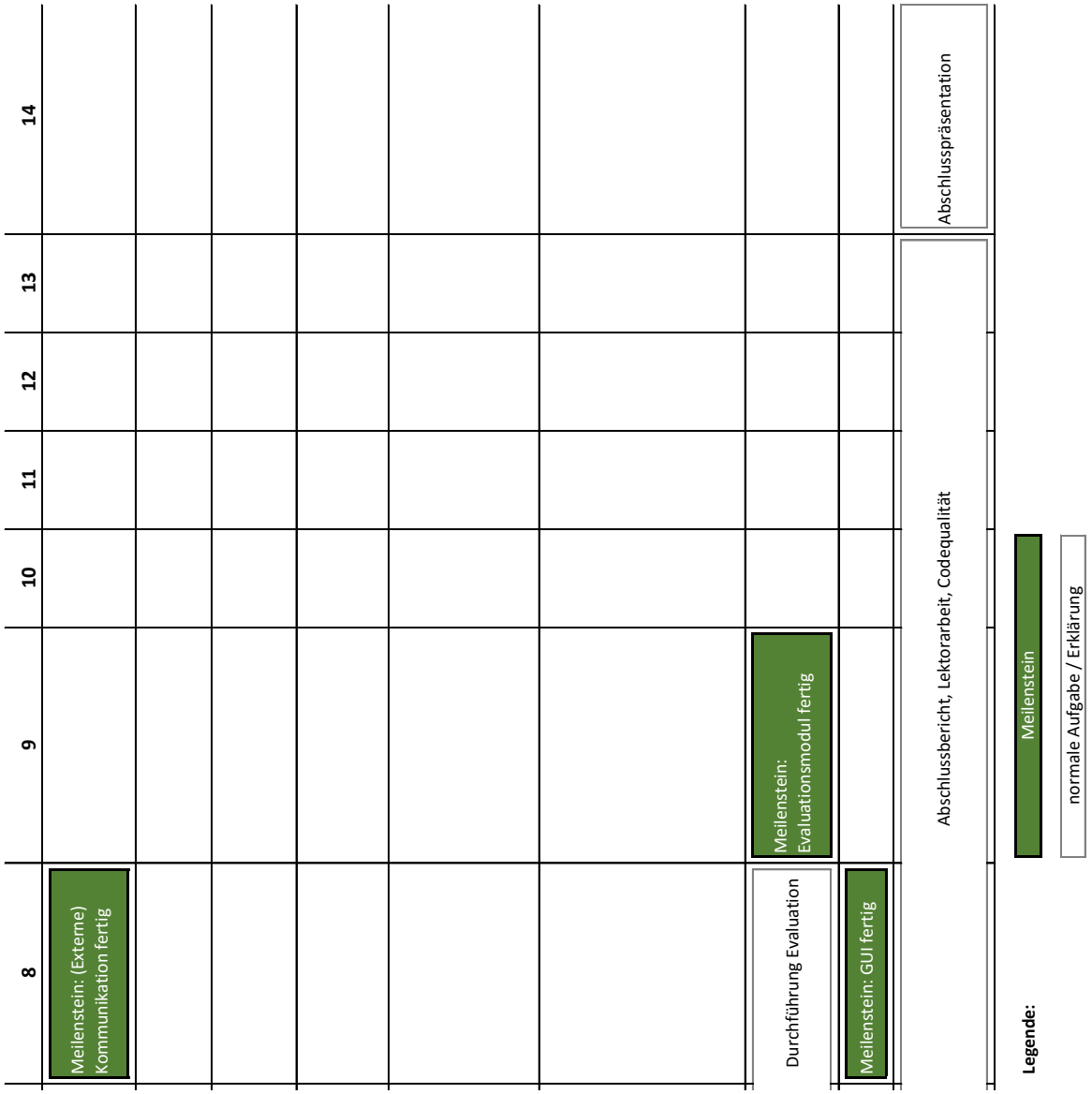


Abbildung 66: Netzplan des Projekts (6/6)

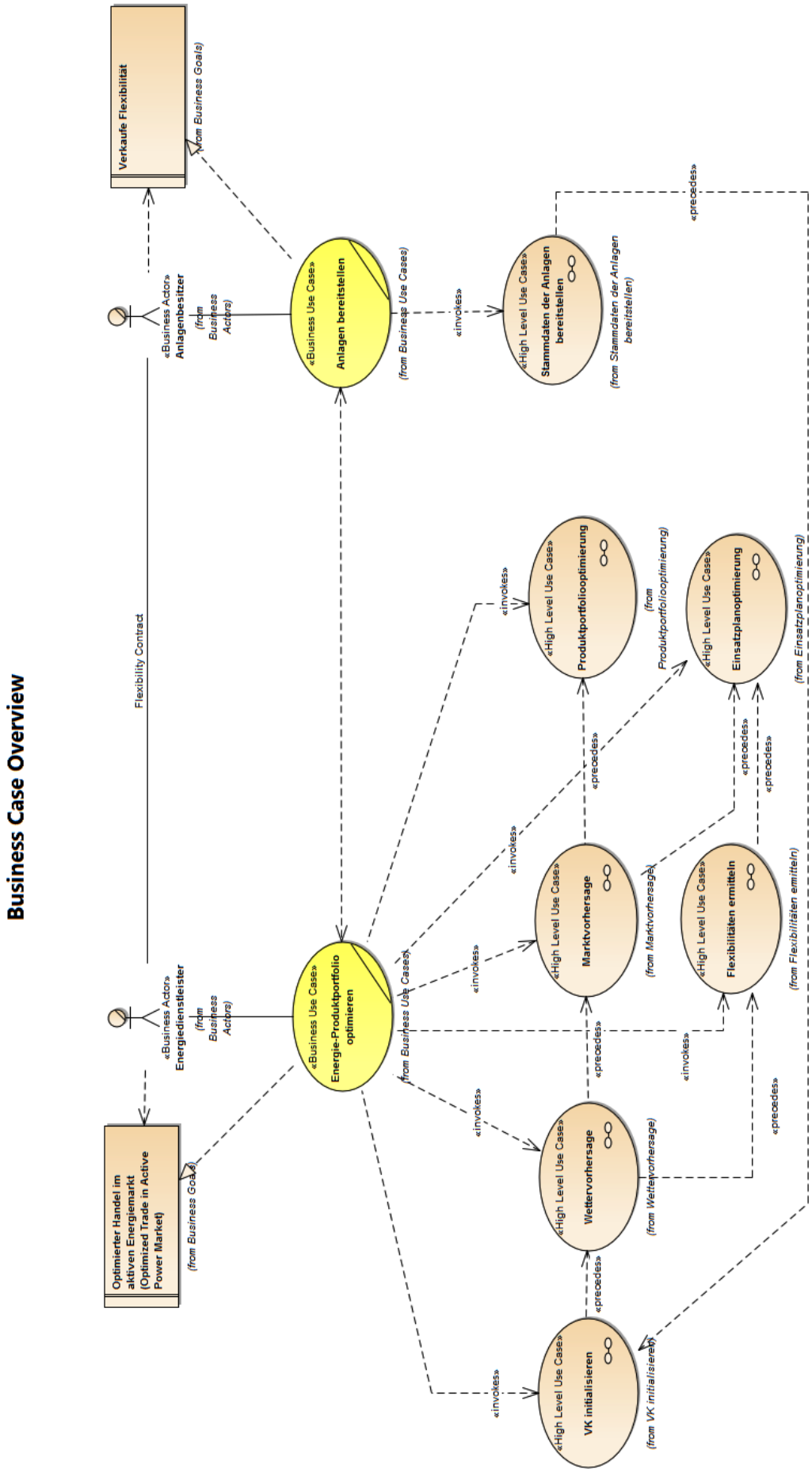


Abbildung 67: Business Use Cases

D. SGAM-Modelle

D.1. SGAM Function Layer

D.1.1. HLUC *VK Initalisieren*

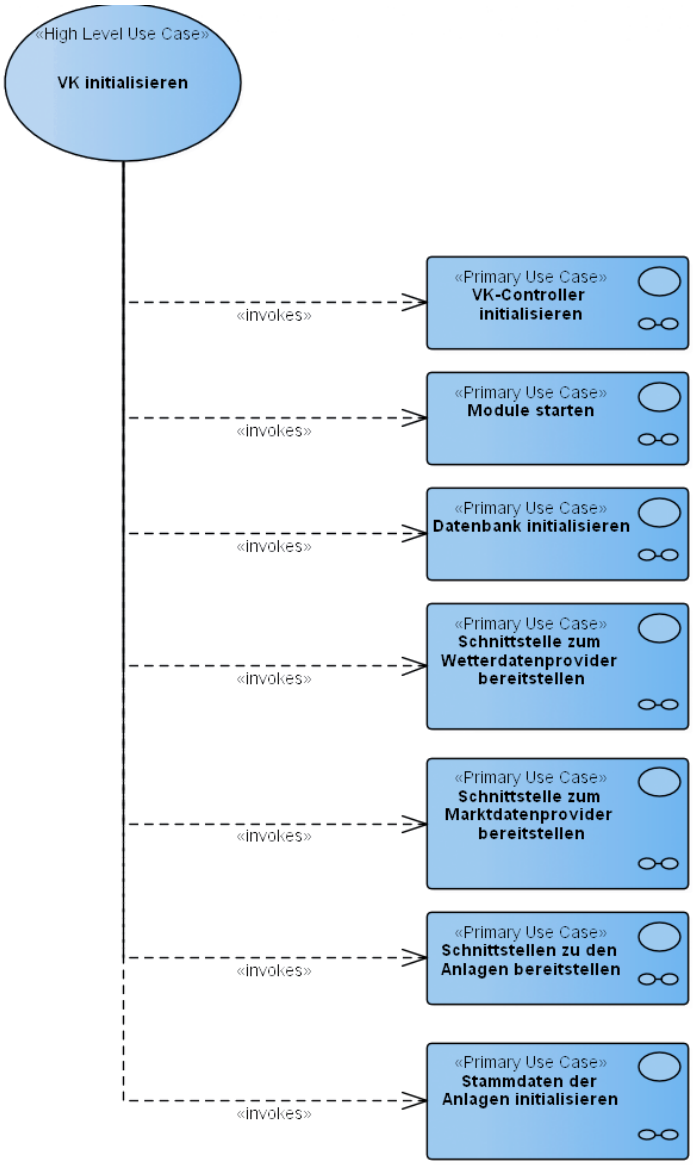
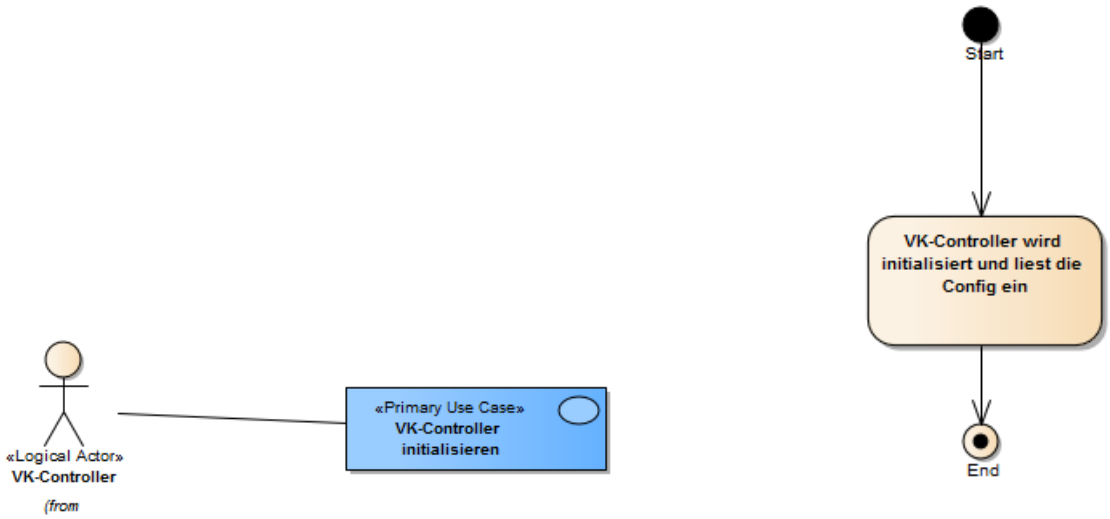
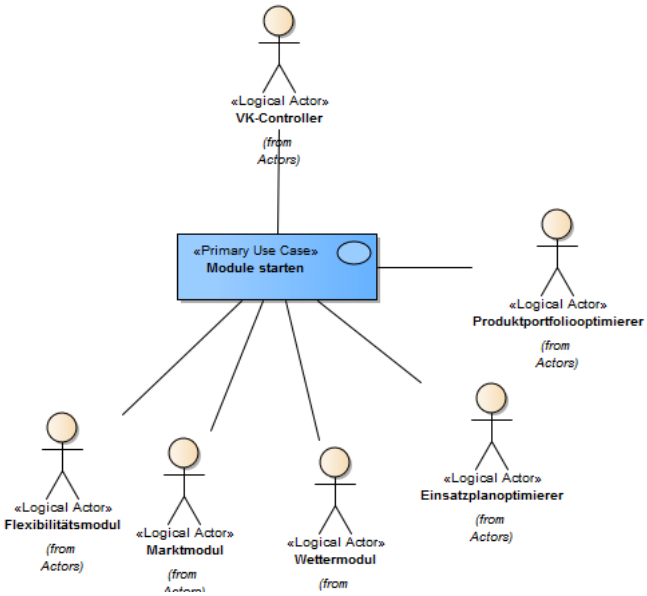


Abbildung 68: HLUC *Virtuelles Kraftwerk initialisieren*

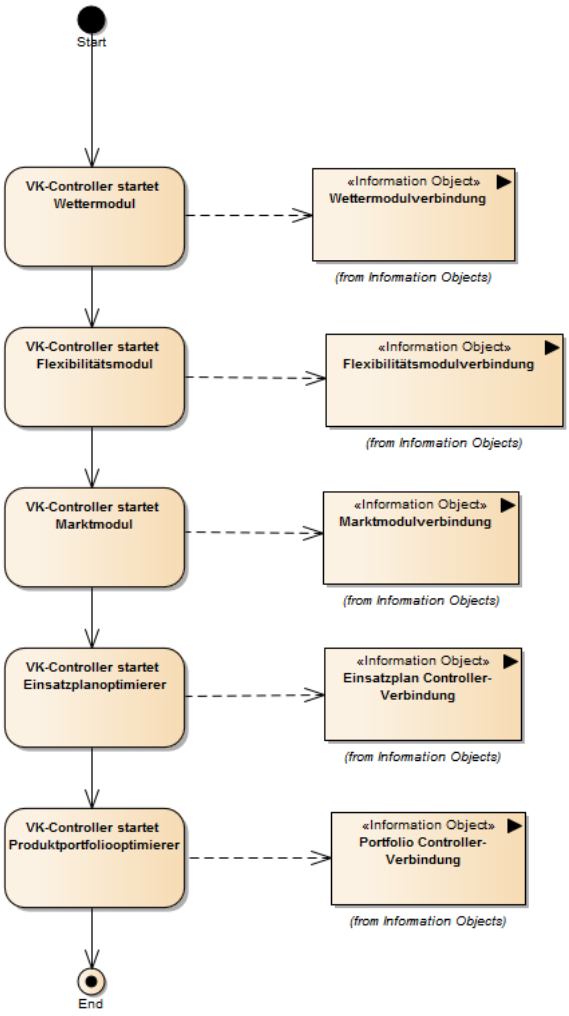


(a) Akteure (b) Aktivitätsdiagramm

Abbildung 69: PUC *VK-Controller initialisieren*

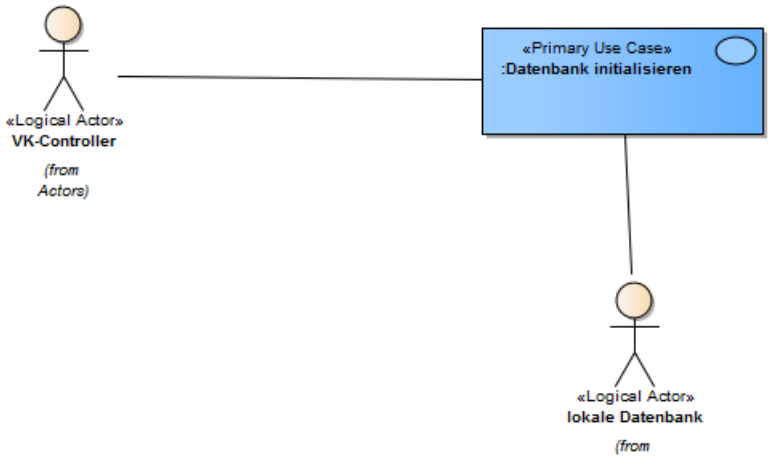


(a) Akteure

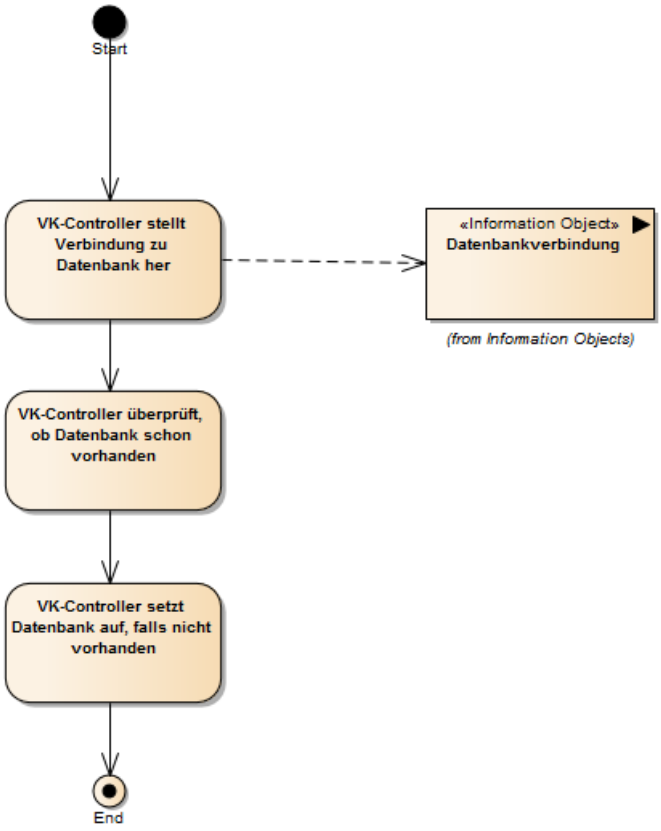


(b) Aktivitätsdiagramm

Abbildung 70: PUC *Module starten*

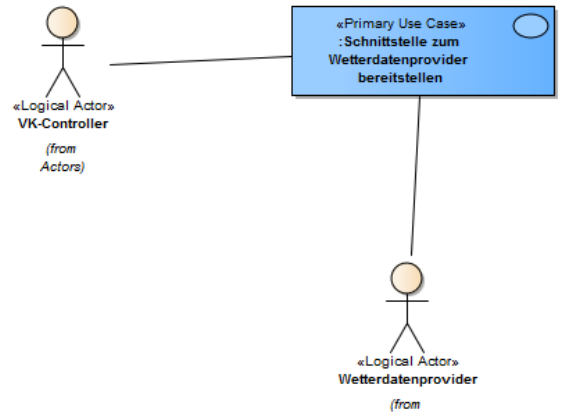


(a) Akteure

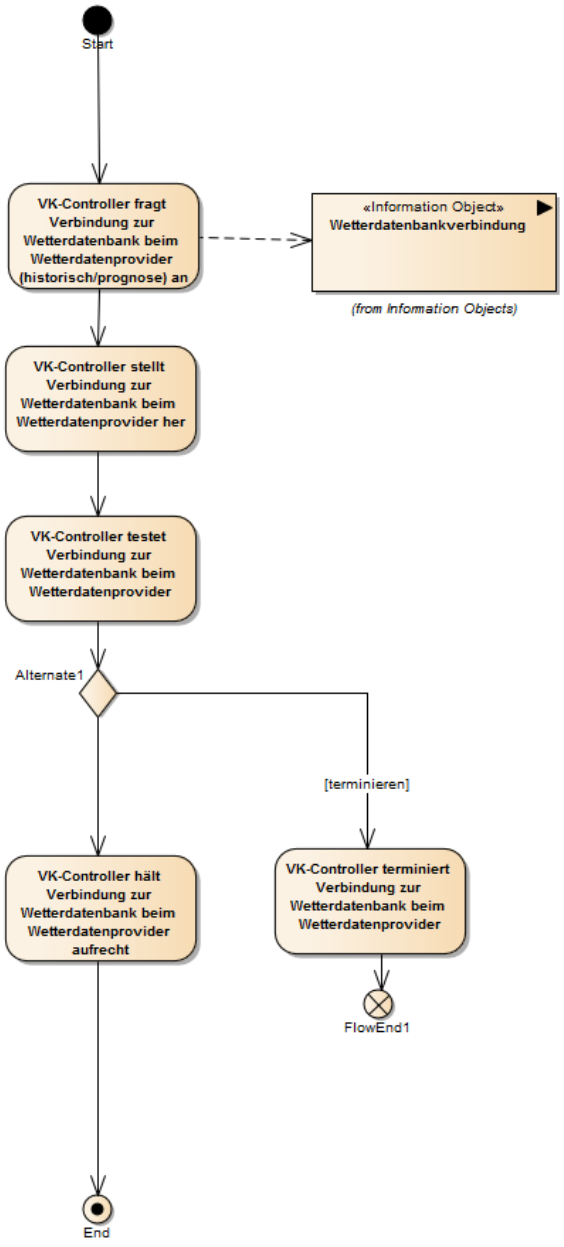


(b) Aktivitätsdiagramm

Abbildung 71: PUC *Lokale Datenbank initialisieren*

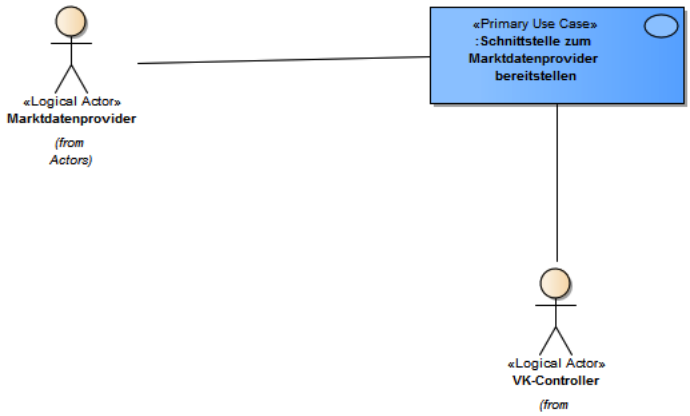


(a) Akteure

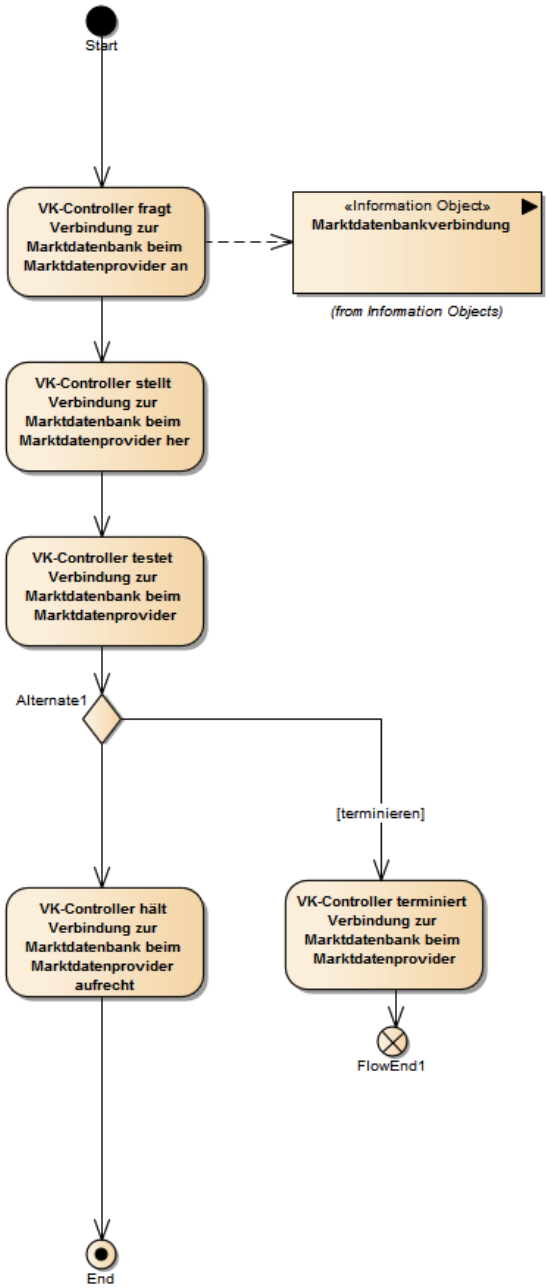


(b) Aktivitätsdiagramm

Abbildung 72: PUC *Schnittstelle zum Wetterdatenprovider bereitstellen*

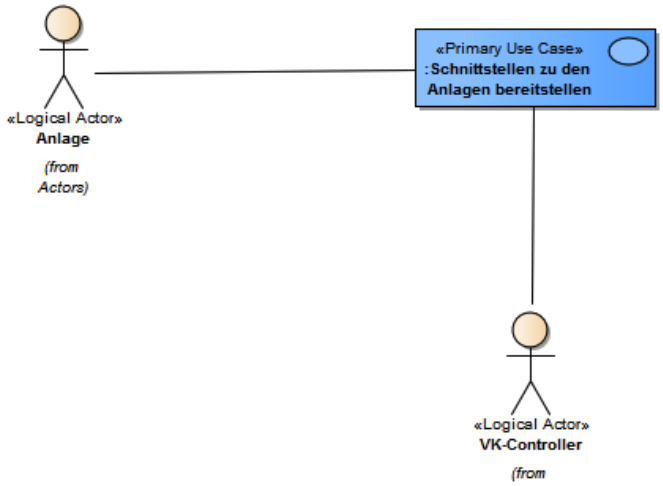


(a) Akteure

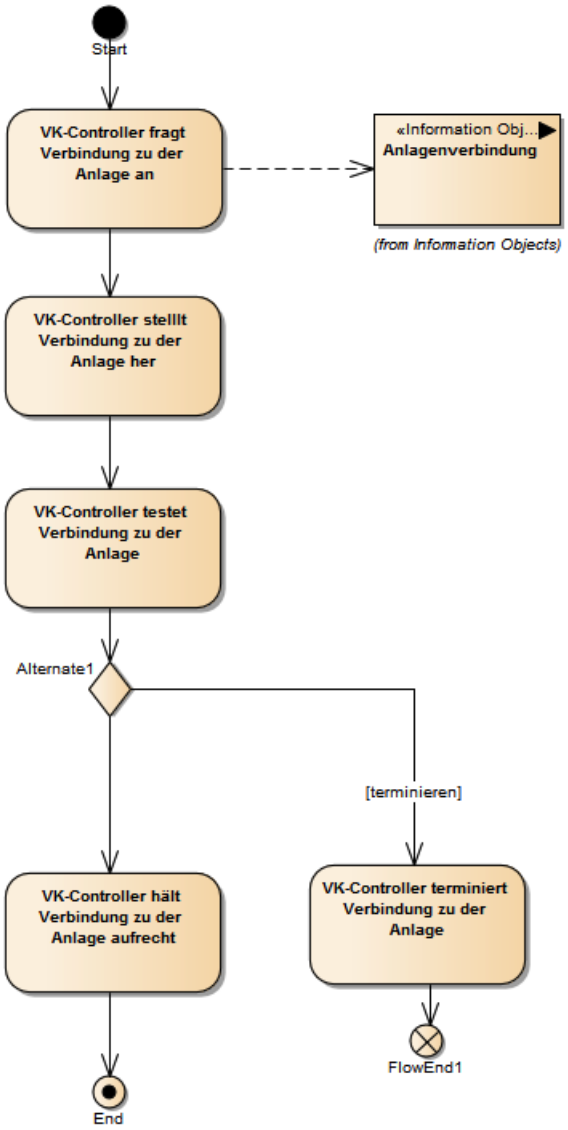


(b) Aktivitätsdiagramm

Abbildung 73: PUC *Schnittstelle zum Marktdatenprovider bereitstellen*

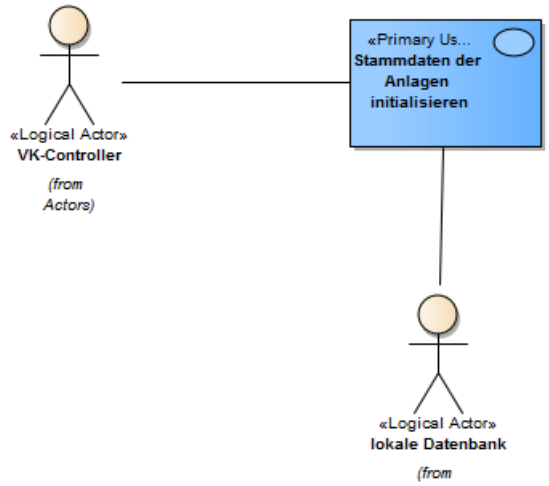


(a) Akteure

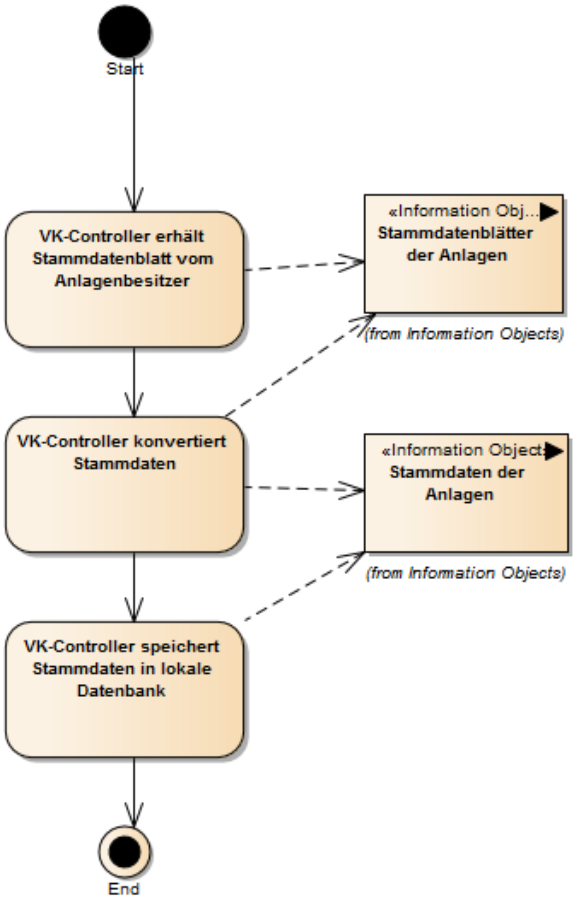


(b) Aktivitätsdiagramm

Abbildung 74: PUC *Schnittstelle zu den Anlagen bereitstellen*



(a) Akteure



(b) Aktivitätsdiagramm

Abbildung 75: PUC *Stammdaten der Anlagen initialisieren*

D.1.2. HLUC *Wetterprognose erstellen*

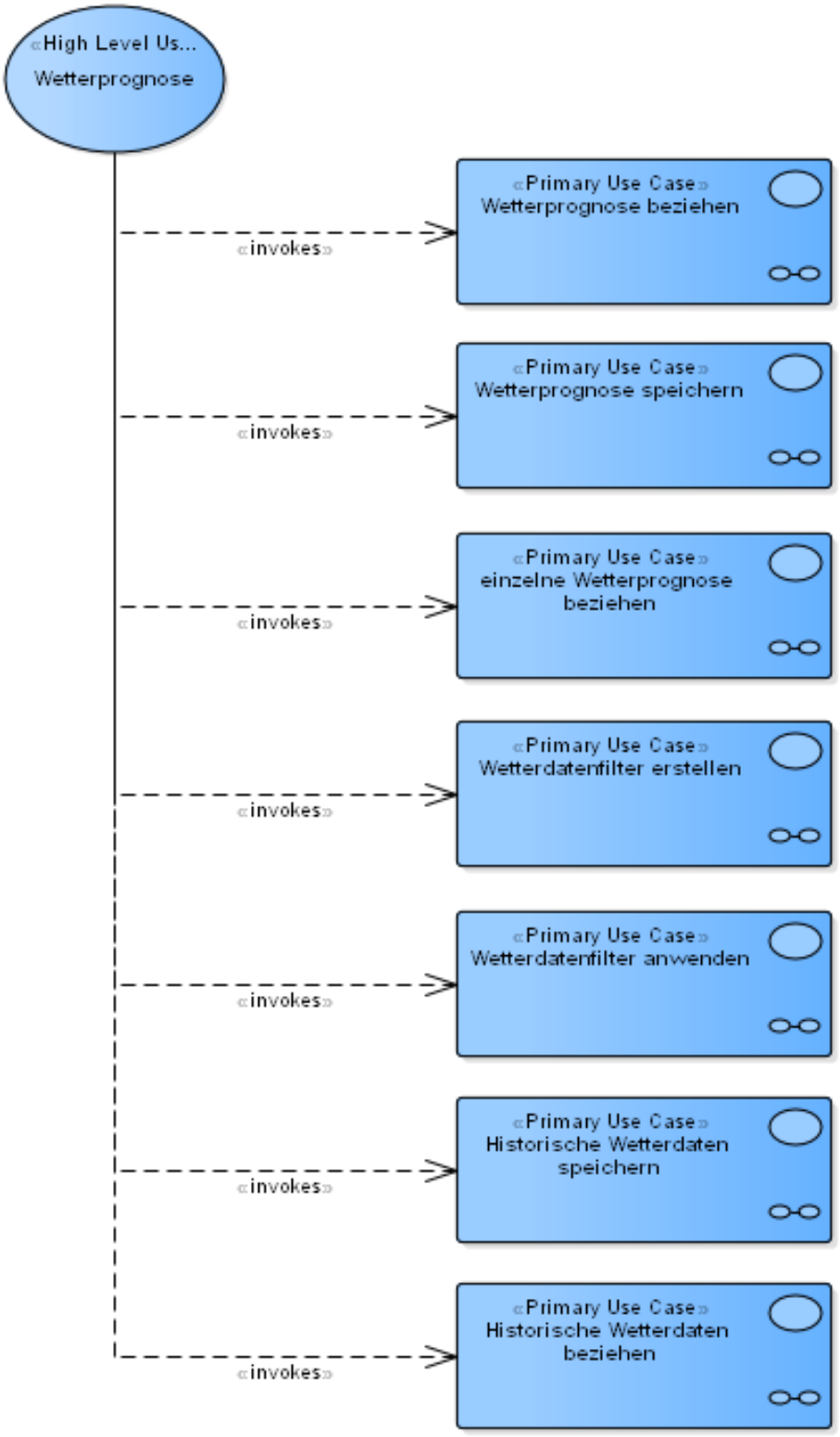
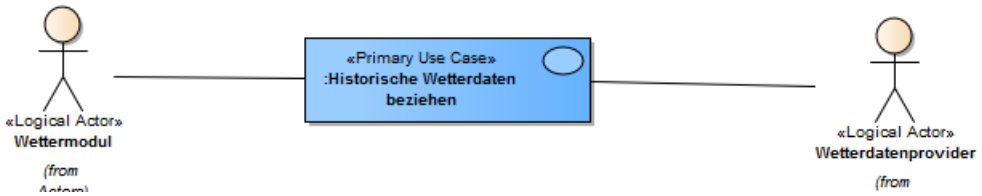
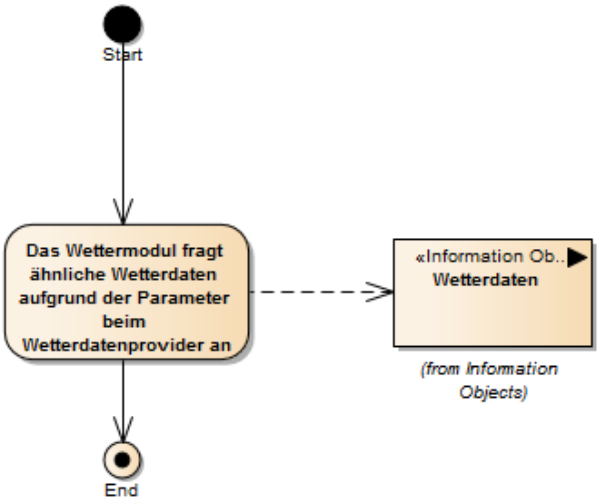


Abbildung 76: HLUC *Wettervorhersage*

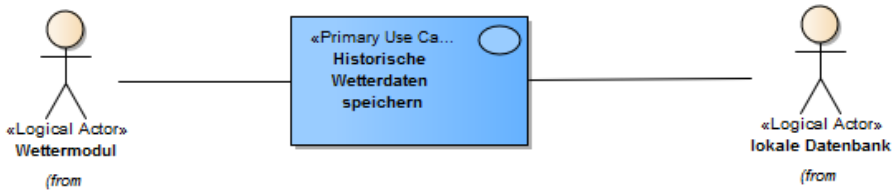


(a) Akteure

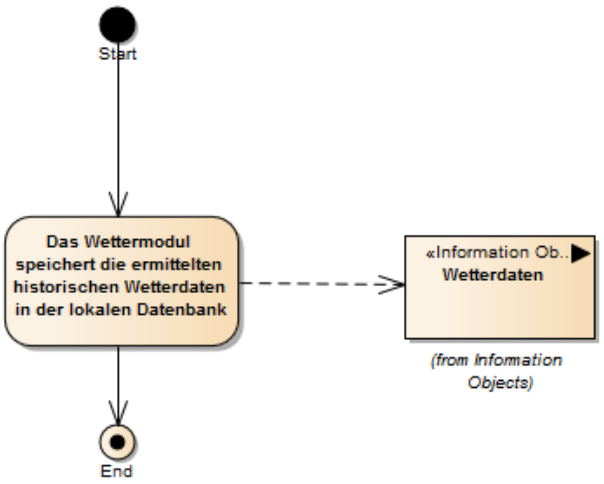


(b) Aktivitätsdiagramm

Abbildung 77: PUC *Historische Wetterdaten beziehen*

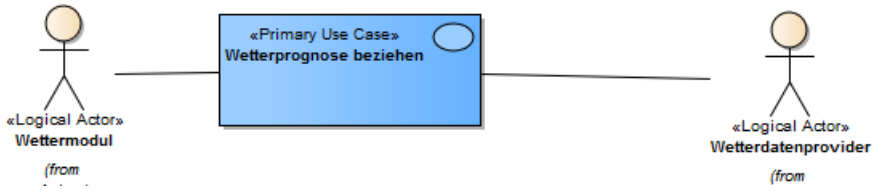


(a) Akteure

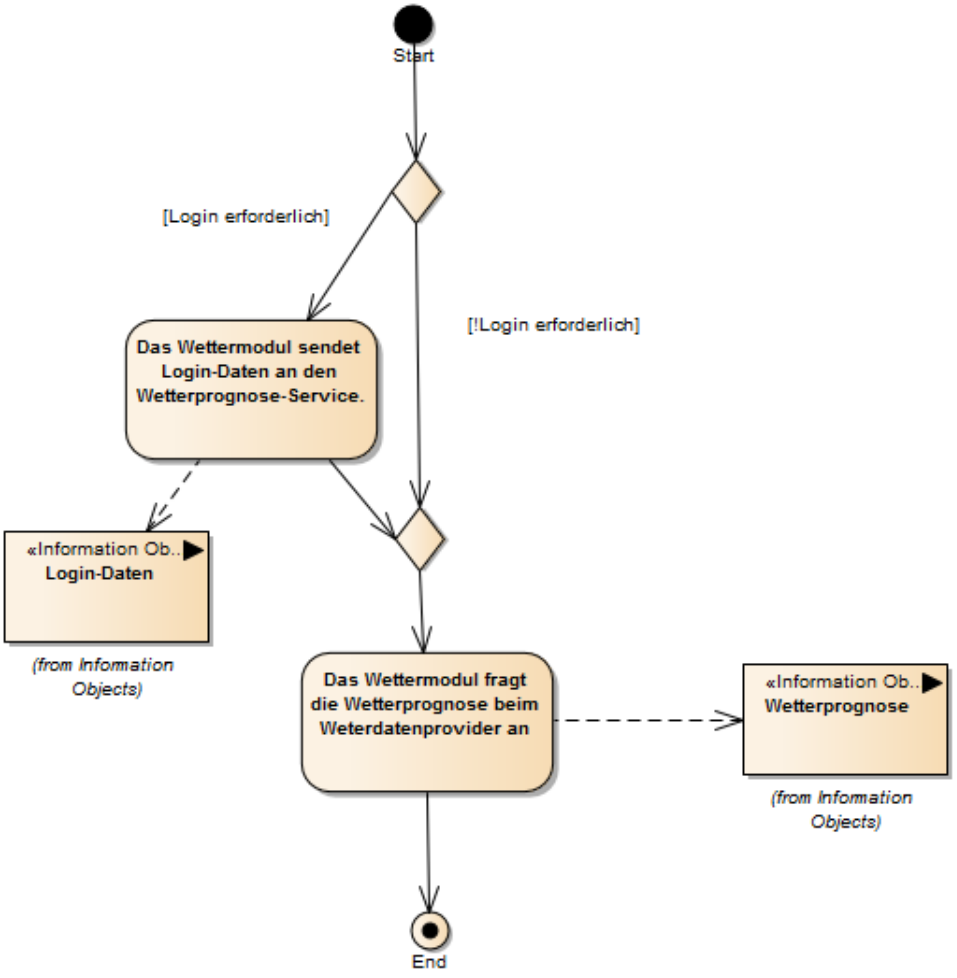


(b) Aktivitätsdiagramm

Abbildung 78: PUC *Historische Wetterdaten speichern*

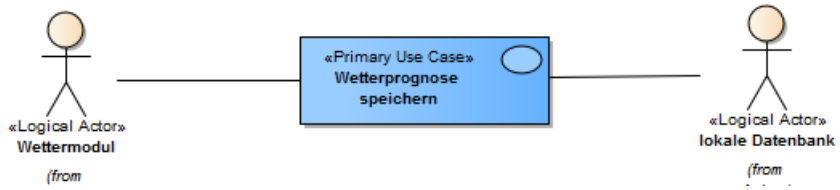


(a) Akteure

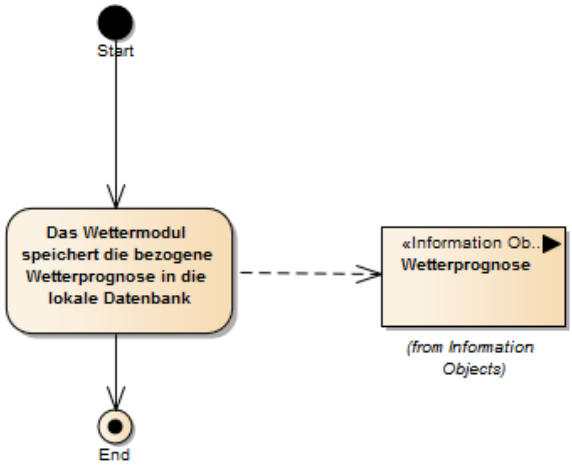


(b) Aktivitätsdiagramm

Abbildung 79: PUC *Einzelne Wetterprognose abrufen*

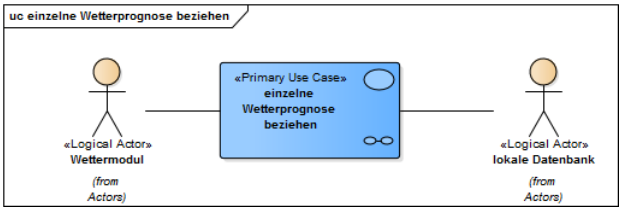


(a) Akteure

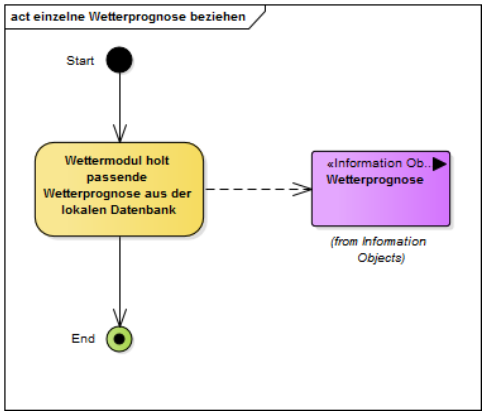


(b) Aktivitätsdiagramm

Abbildung 80: PUC *Wetterprognose speichern*

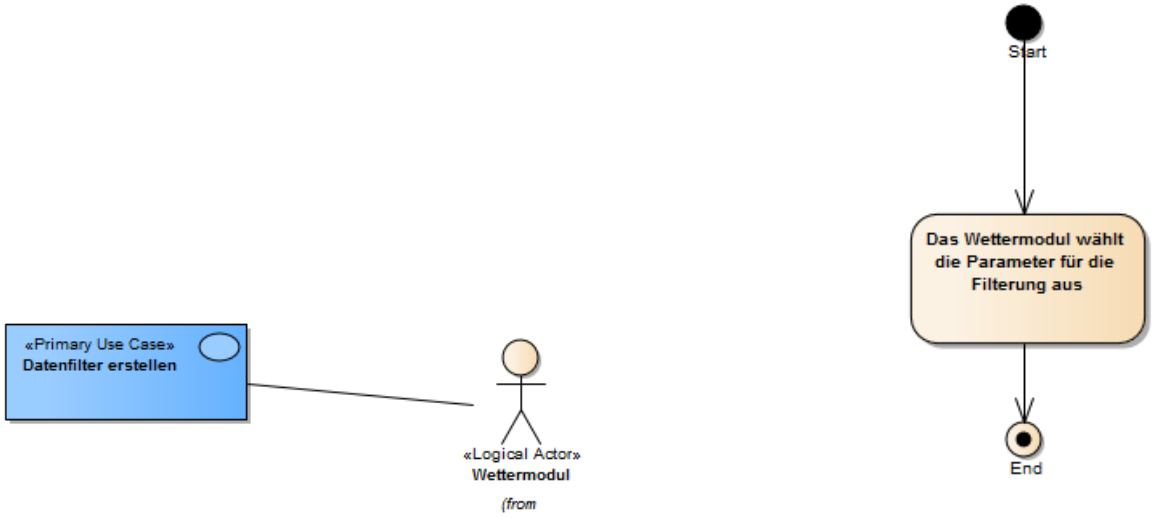


(a) Akteure



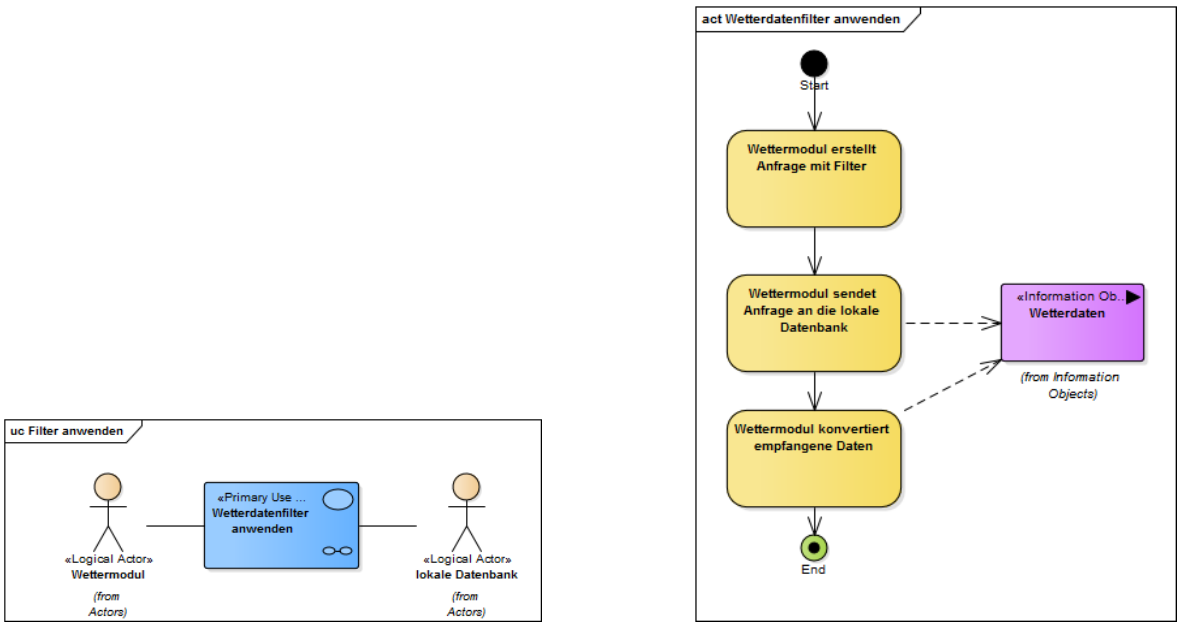
(b) Aktivitätsdiagramm

Abbildung 81: PUC *einzelne Wetterprognose beziehen*



(a) Akteure (b) Aktivitätsdiagramm

Abbildung 82: PUC *Datenfilter erstellen*



(a) Akteure (b) Aktivitätsdiagramm

Abbildung 83: PUC *Datenfilter anwenden*

D.1.3. HLUC *Marktprognose erstellen*

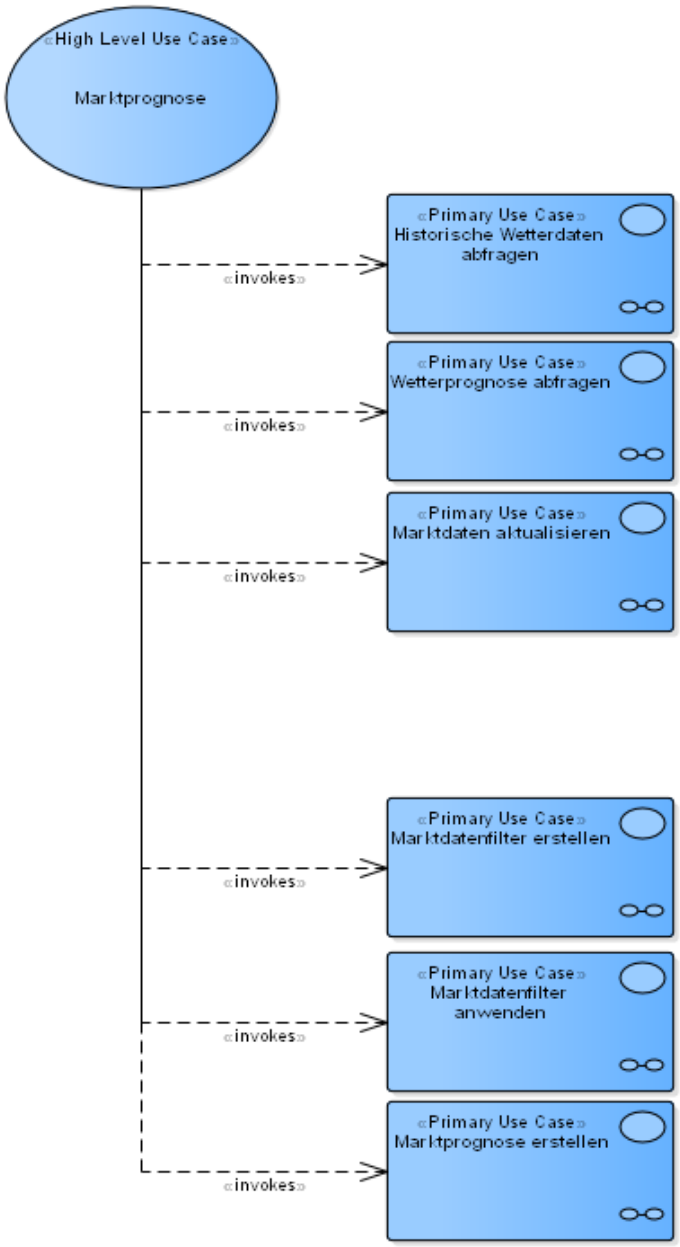
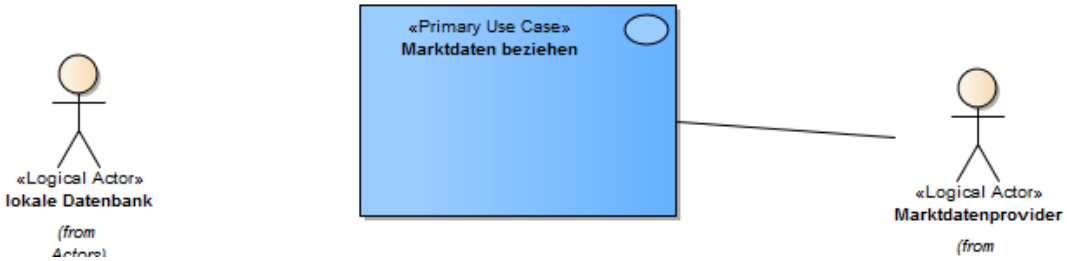
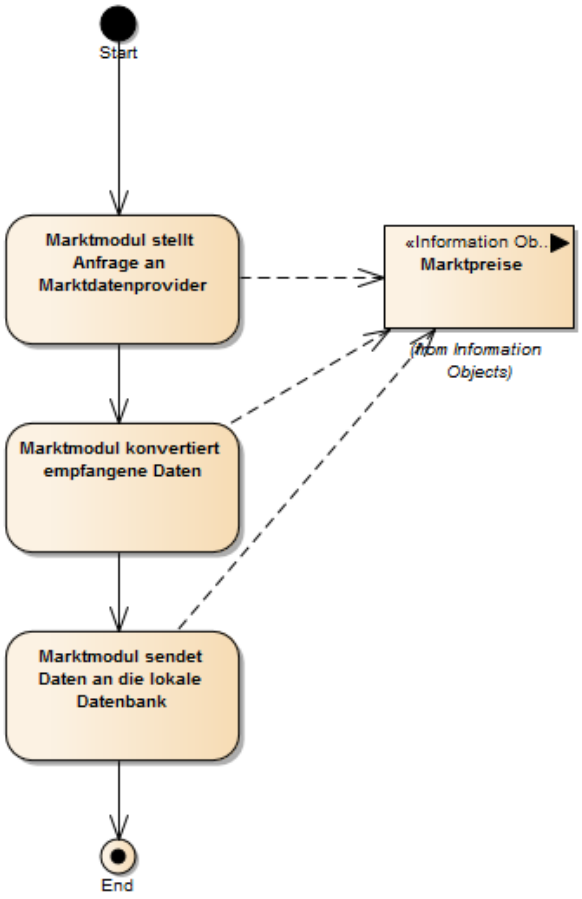


Abbildung 84: HLUC *Marktprognose*

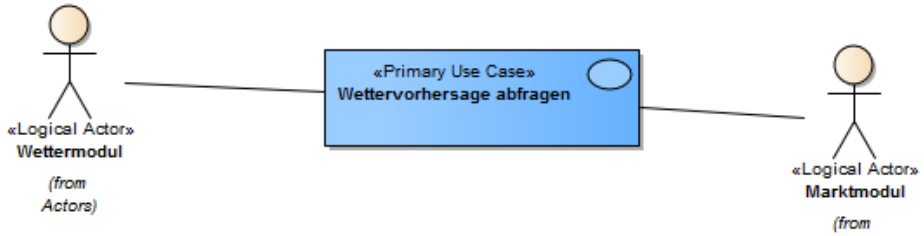


(a) Akteure

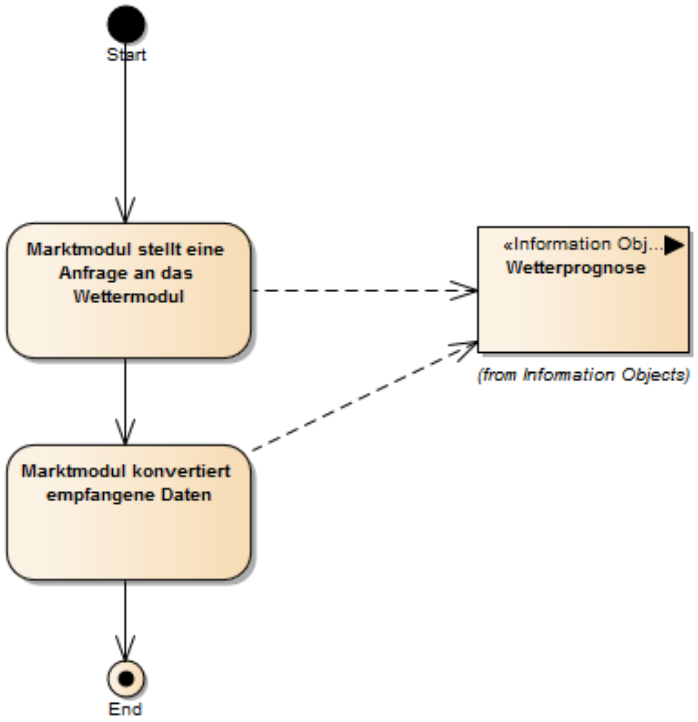


(b) Aktivitätsdiagramm

Abbildung 85: PUC *Marktdaten aktualisieren*

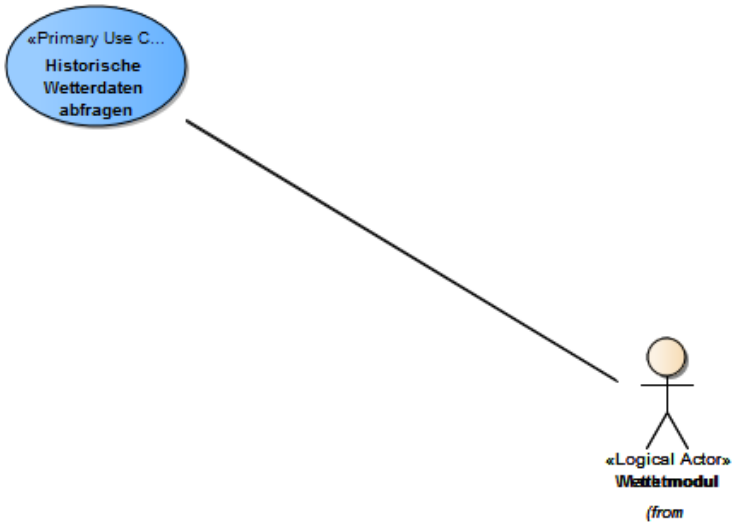


(a) Akteure

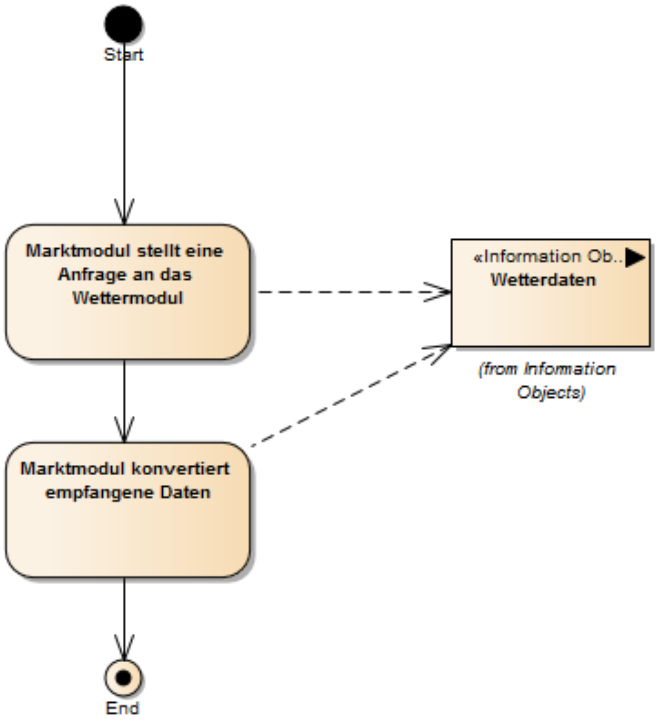


(b) Aktivitätsdiagramm

Abbildung 86: PUC *Wettervorhersage beziehen*

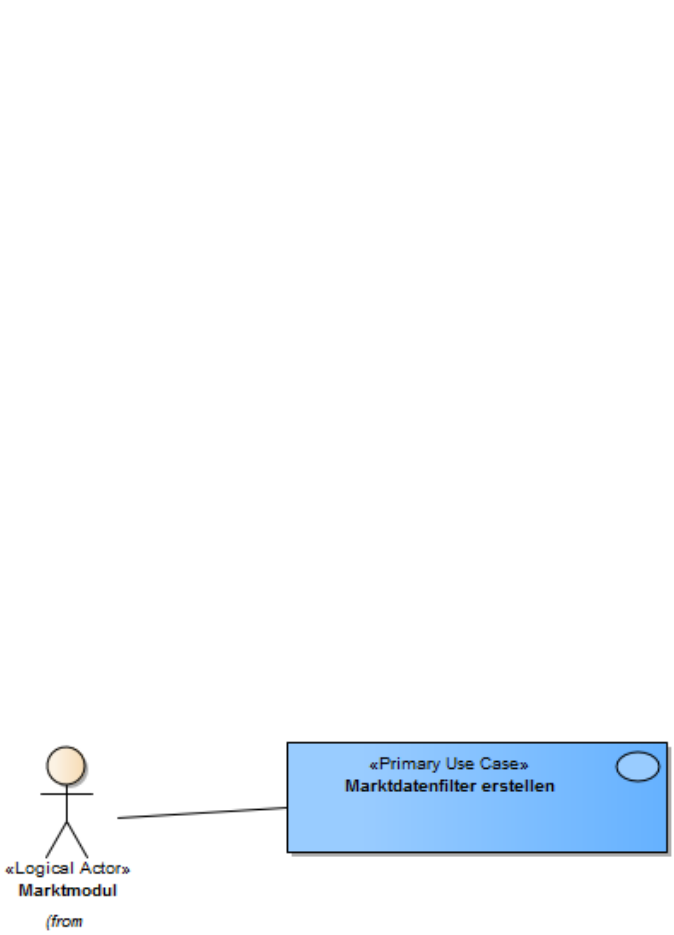


(a) Akteure

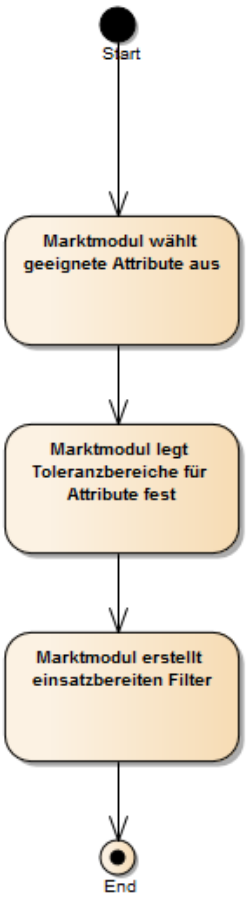


(b) Aktivitätsdiagramm

Abbildung 87: PUC *Historische Wetterdaten abfragen*

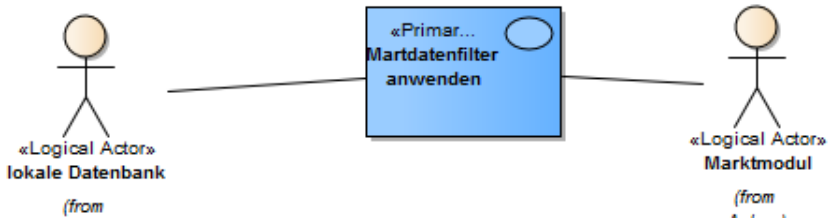


(a) Akteure

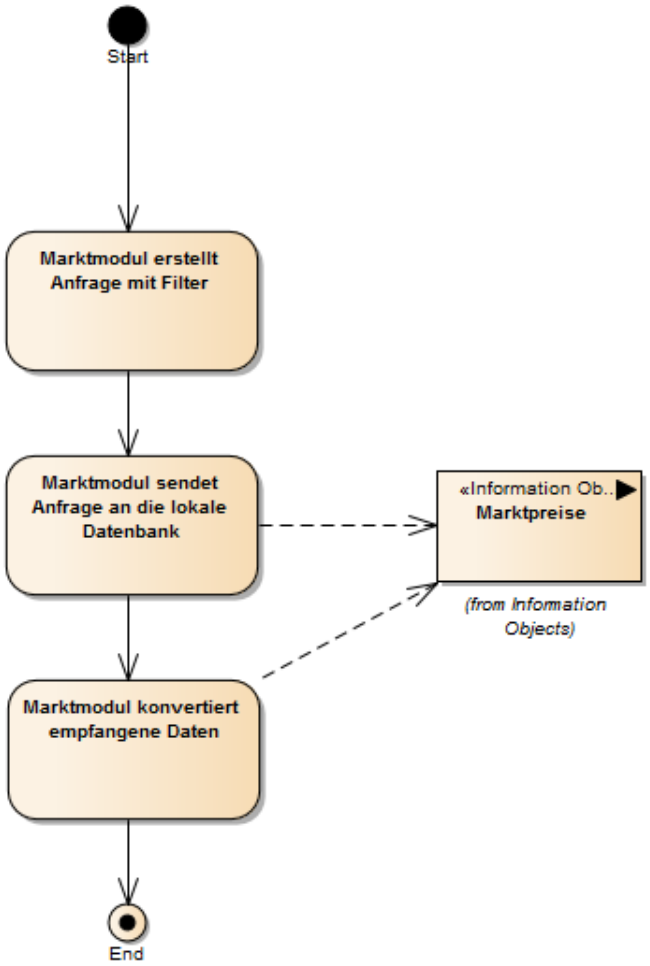


(b) Aktivitätsdiagramm

Abbildung 88: PUC *Marktdatenfilter erstellen*

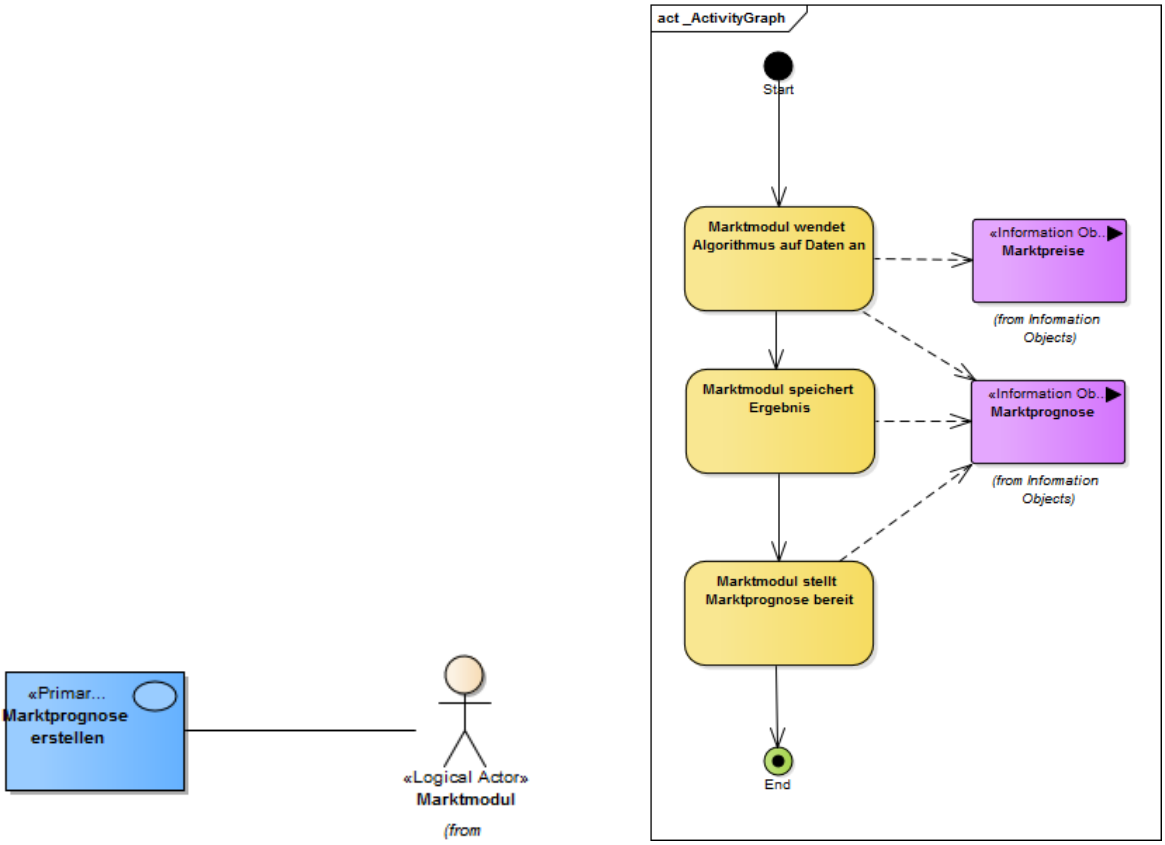


(a) Akteure



(b) Aktivitätsdiagramm

Abbildung 89: PUC *Marktdatenfilter anwenden*



(a) Akteure (b) Aktivitätsdiagramm

Abbildung 90: PUC *Marktprognose erstellen*

D.1.4. HLUC *Flexibilitäten ermitteln*

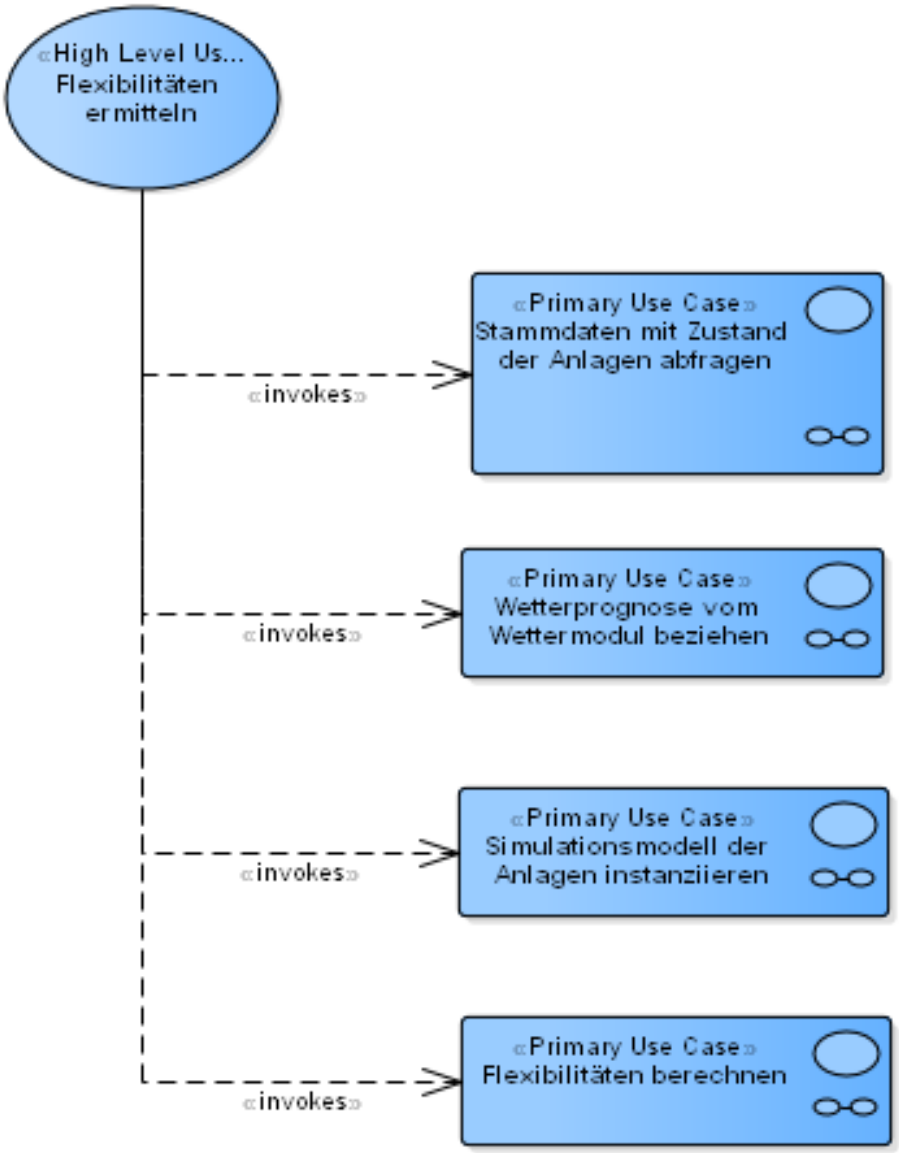
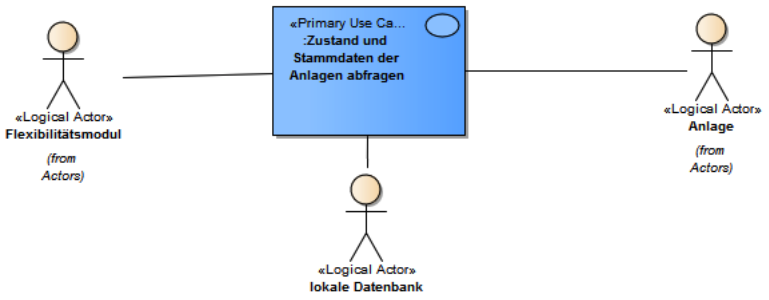
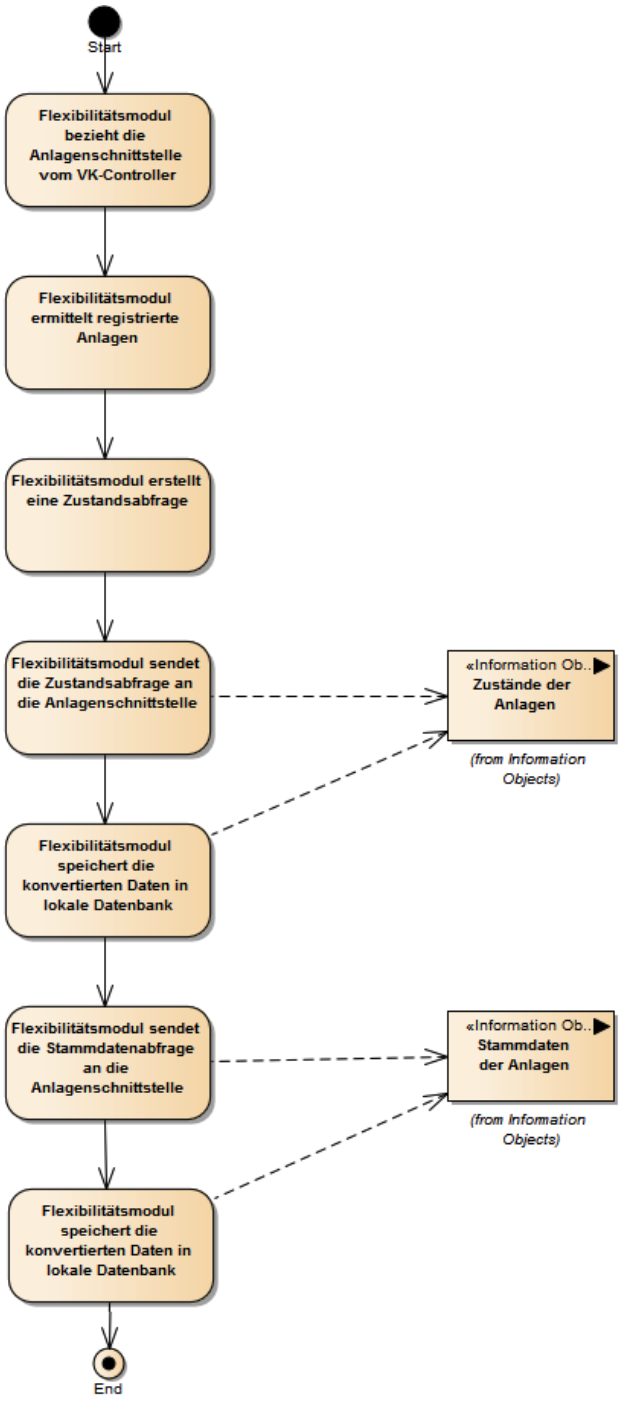


Abbildung 91: HLUC *Flexibilitäten ermitteln*

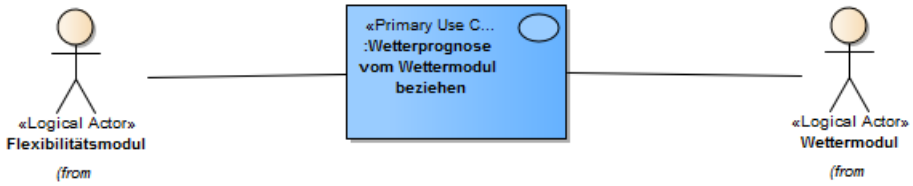


(a) Akteure

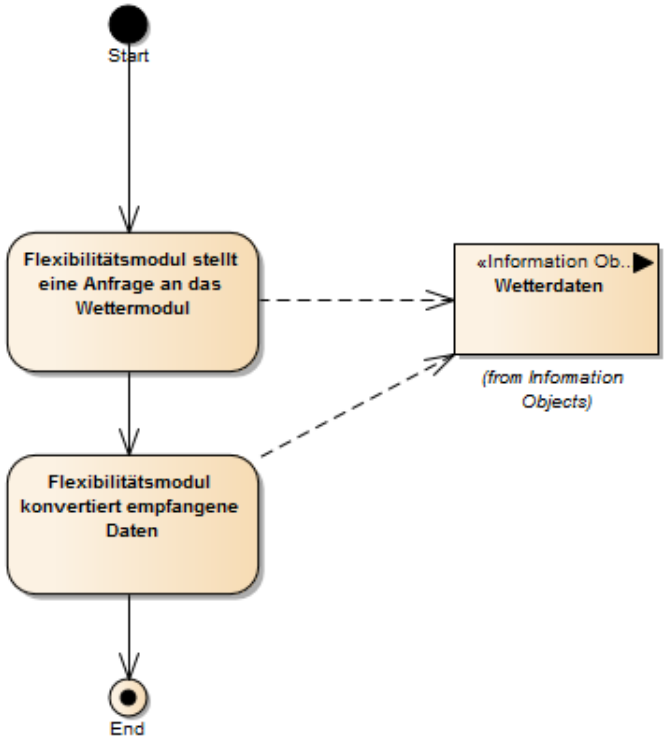


(b) Aktivitätsdiagramm

Abbildung 92: PUC *Stammdaten mit Zustand der Anlage abfragen*

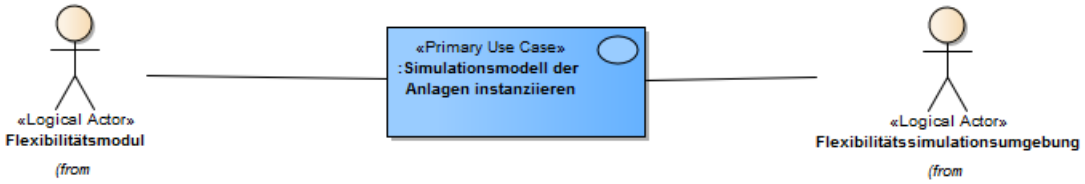


(a) Akteure

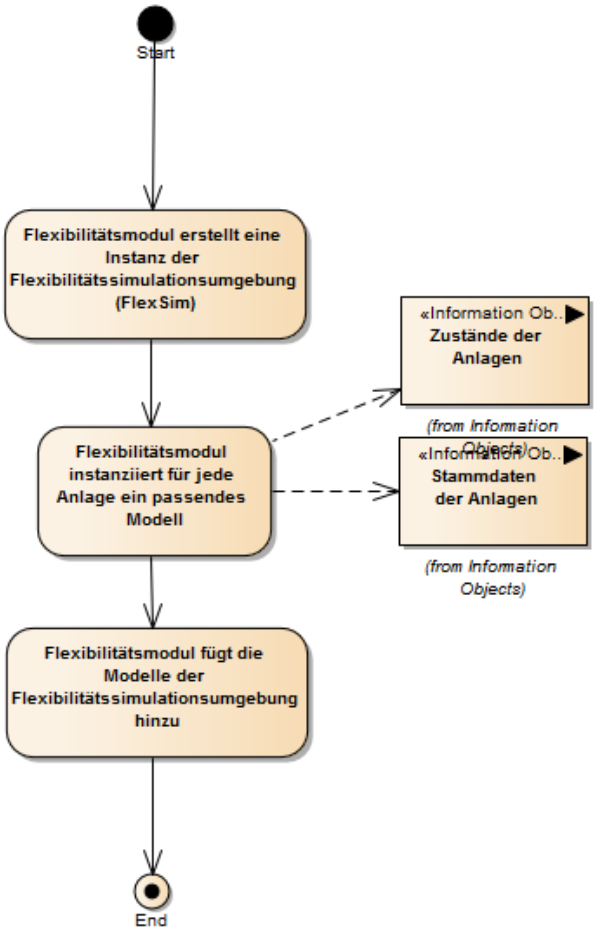


(b) Aktivitätsdiagramm

Abbildung 93: PUC *Wetterprognose vom Wettermodul beziehen*

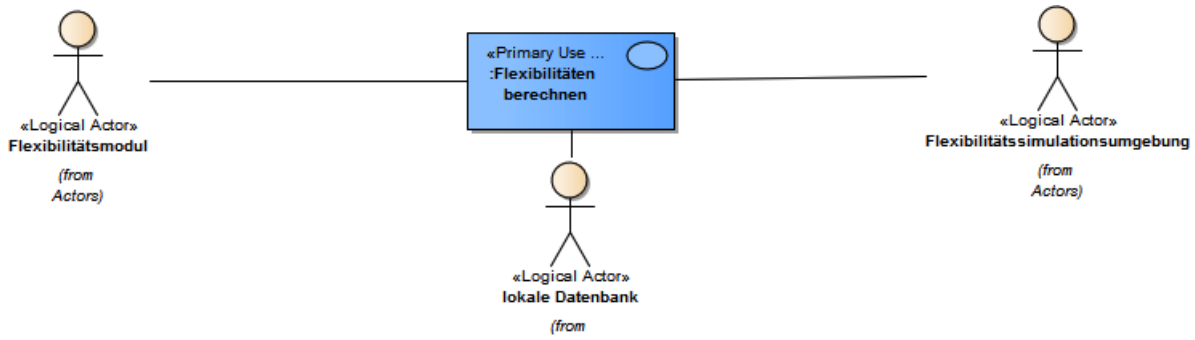


(a) Akteure

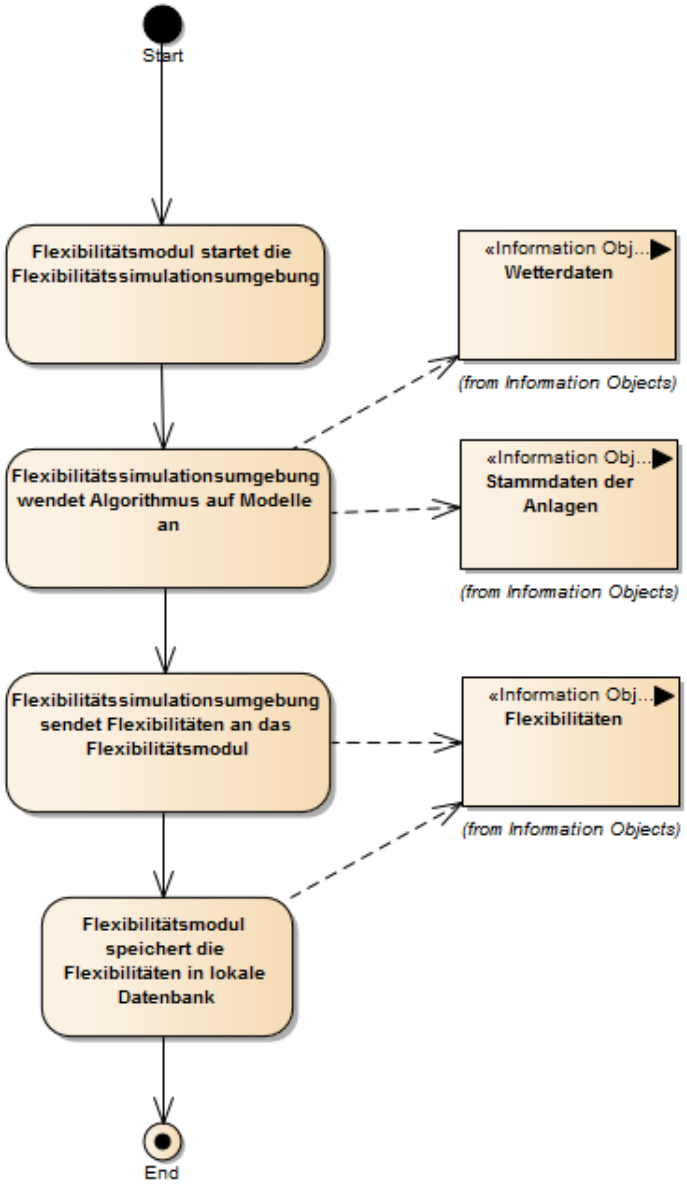


(b) Aktivitätsdiagramm

Abbildung 94: PUC *Simulationsmodell der Anlagen instanziieren*



(a) Akteure



(b) Aktivitätsdiagramm

Abbildung 95: PUC *Flexibilitäten berechnen*

D.1.5. HLUC *Einsatzplanoptimierung*

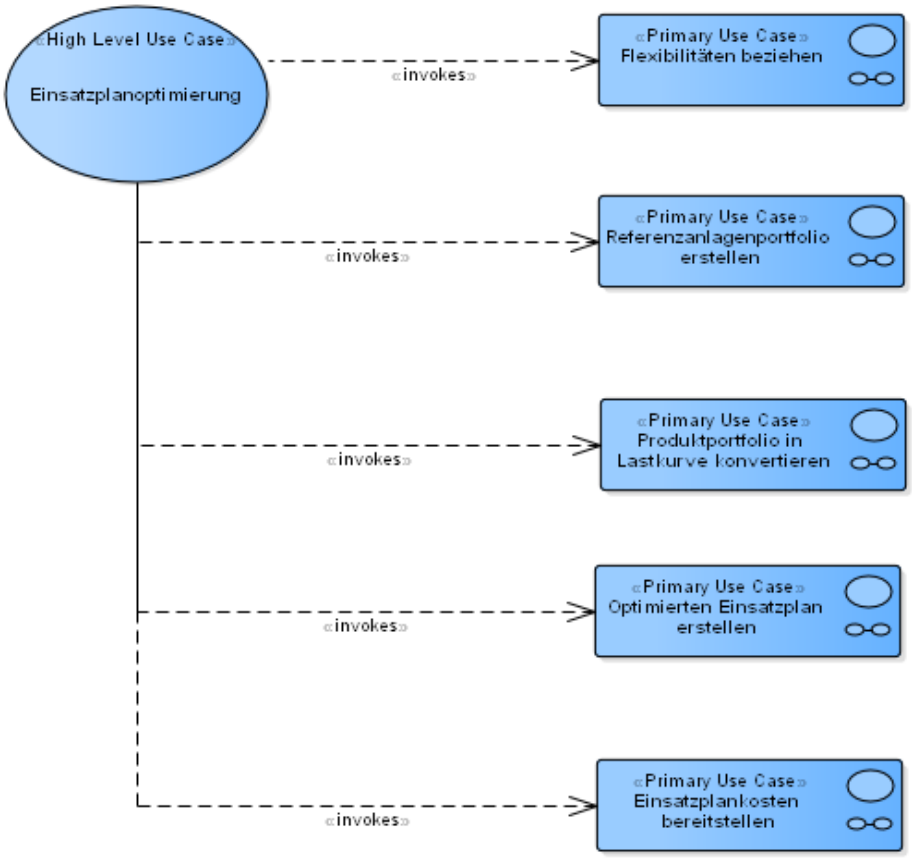
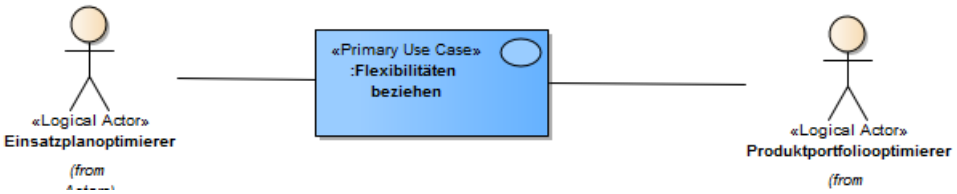
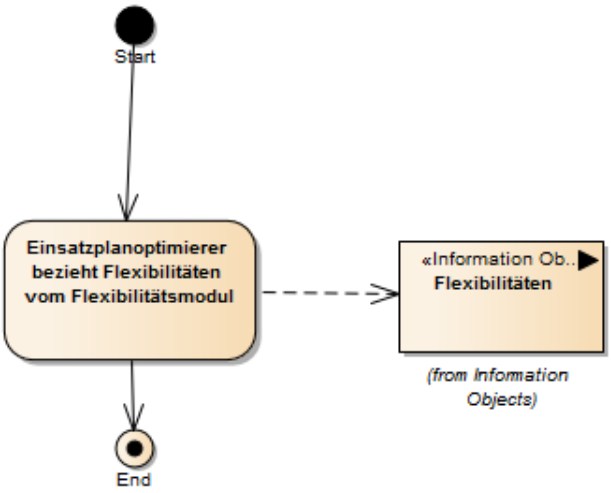


Abbildung 96: HLUC *Einsatzplanoptimierung*

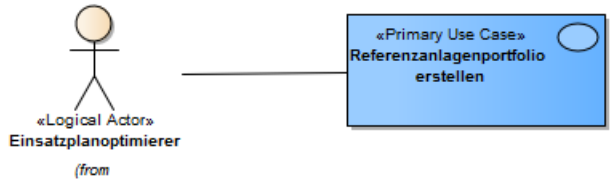


(a) Akteure

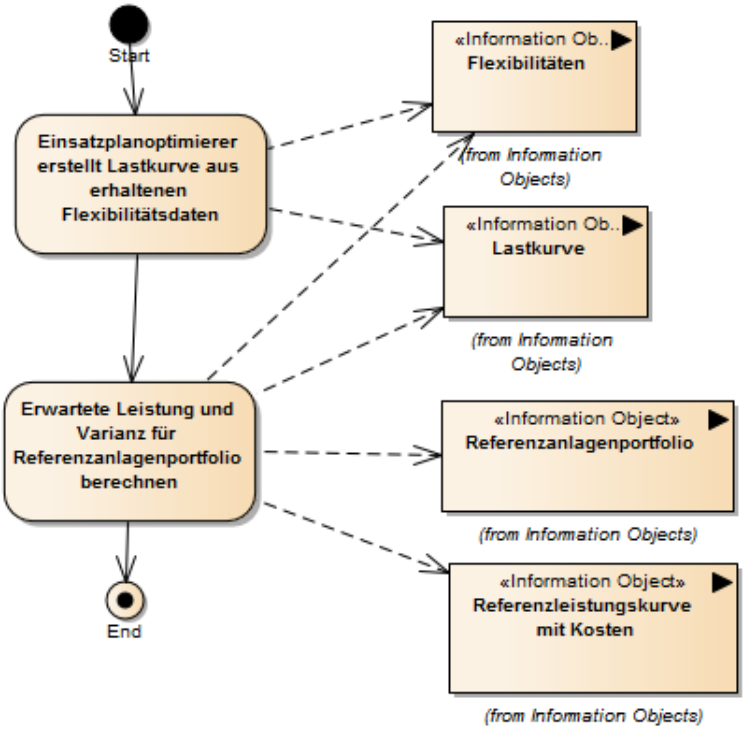


(b) Aktivitätsdiagramm

Abbildung 97: PUC *Flexibilitäten beziehen*

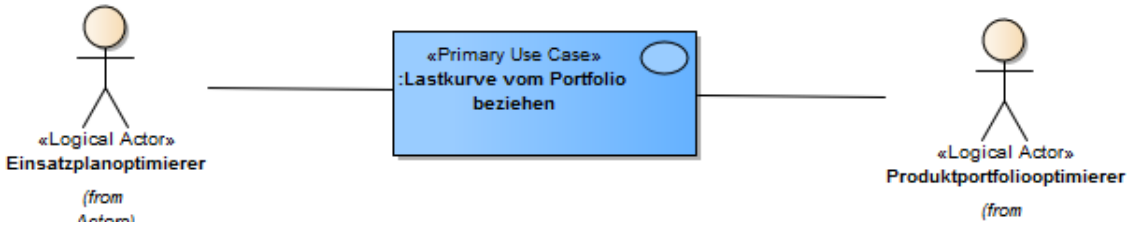


(a) Akteure

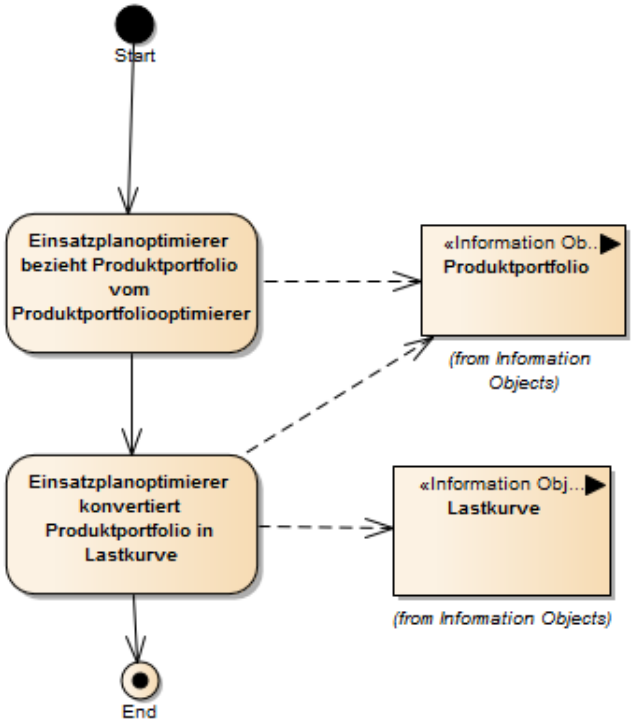


(b) Aktivitätsdiagramm

Abbildung 98: PUC *Referenzanlagenportfolio erstellen*

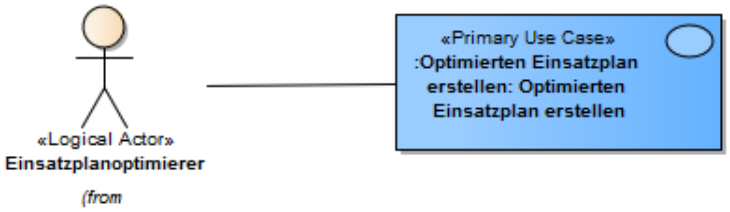


(a) Akteure

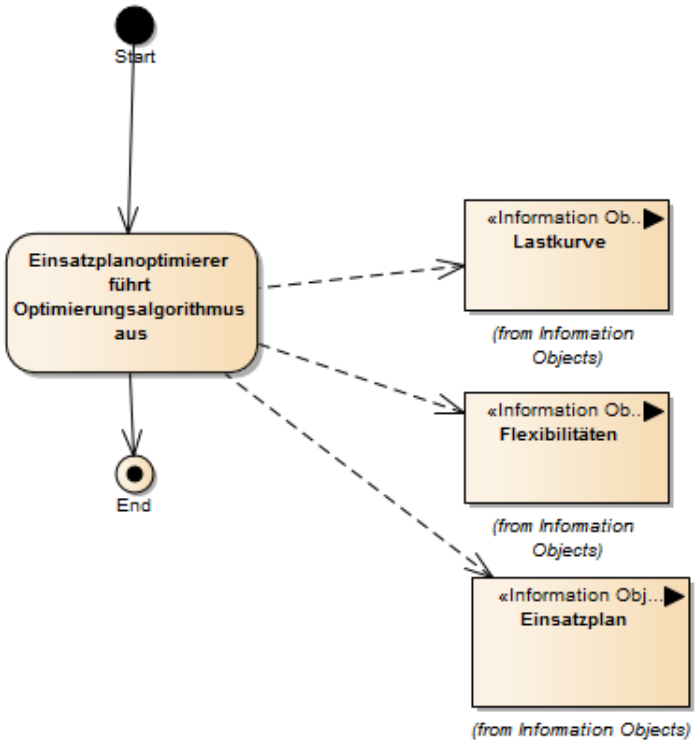


(b) Aktivitätsdiagramm

Abbildung 99: PUC *Lastkurve vom Portfoliooptimierer beziehen*

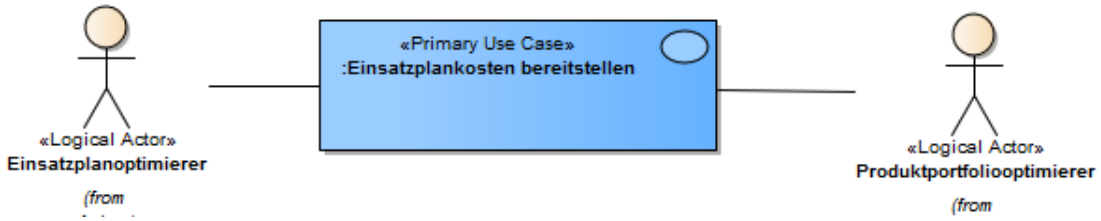


(a) Akteure

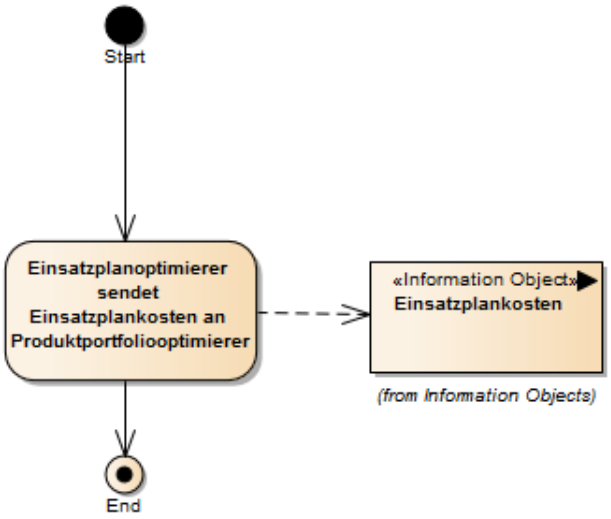


(b) Aktivitätsdiagramm

Abbildung 100: PUC *Optimierten Einsatzplan erstellen*



(a) Akteure



(b) Aktivitätsdiagramm

Abbildung 101: PUC *Einsatzplankosten bereitstellen*

D.1.6. HLUC *Produktportfoliooptimierung*

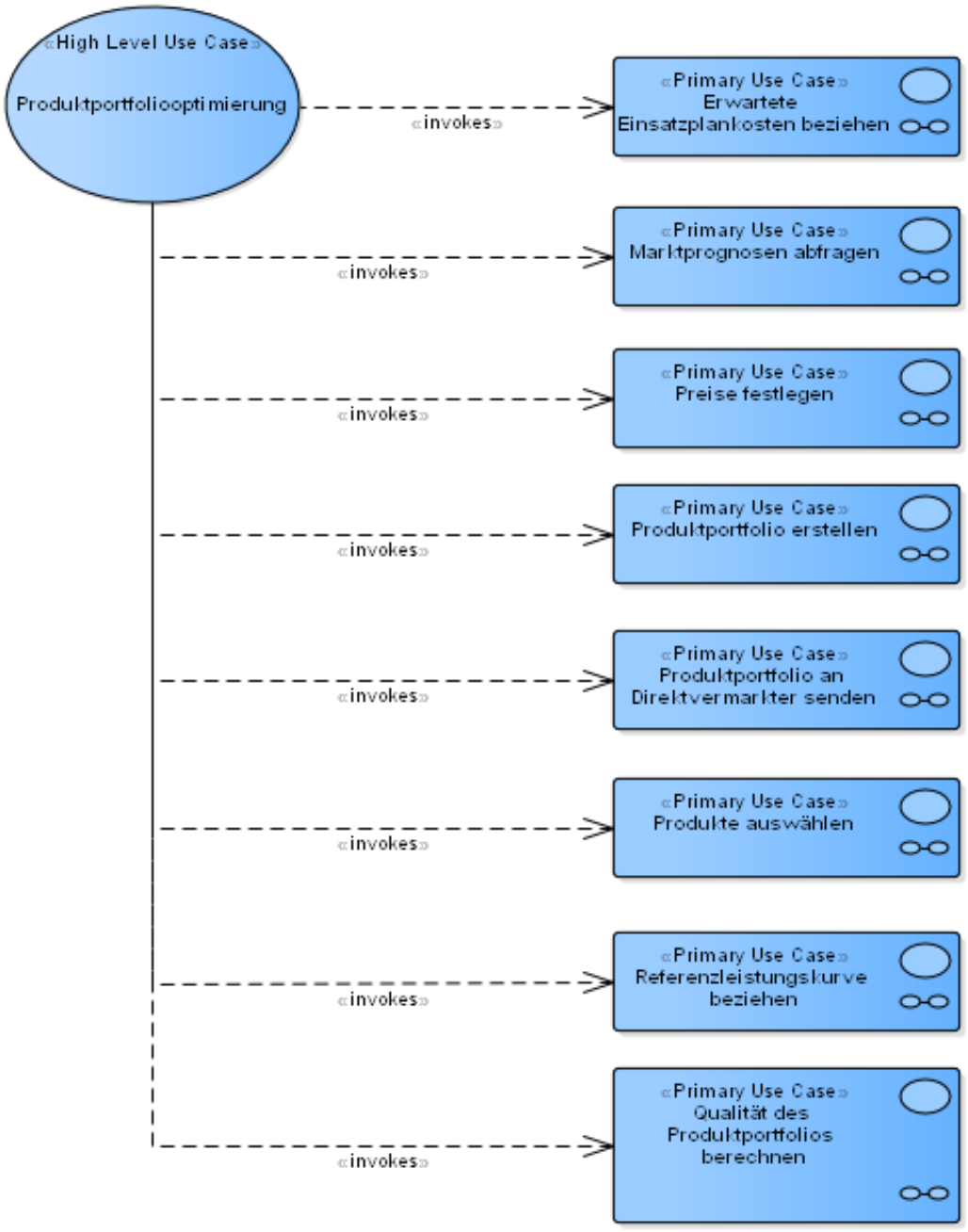
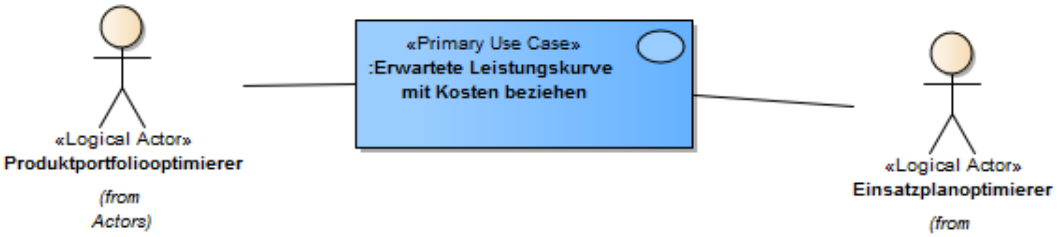
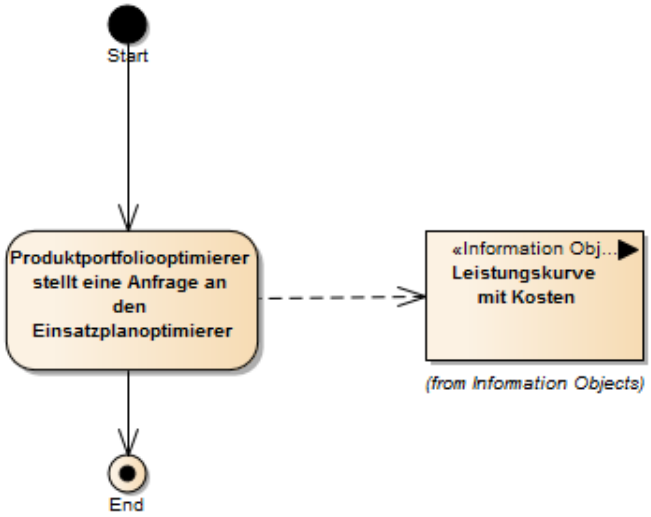


Abbildung 102: HLUC *Produktportfoliooptimierung*

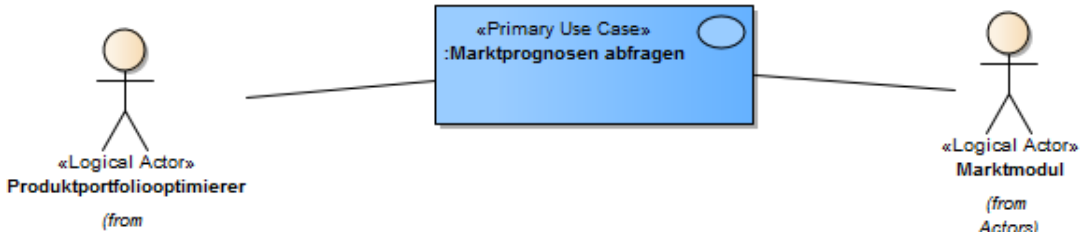


(a) Akteure

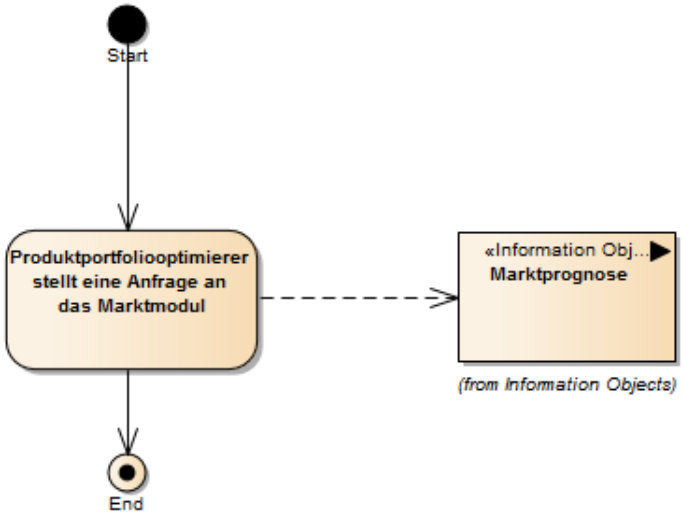


(b) Aktivitätsdiagramm

Abbildung 103: PUC *Erwartete Leistungskurve mit Kosten beziehen*

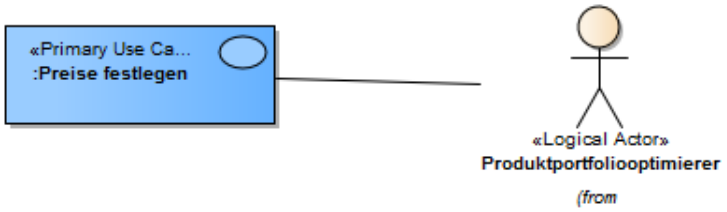


(a) Akteure

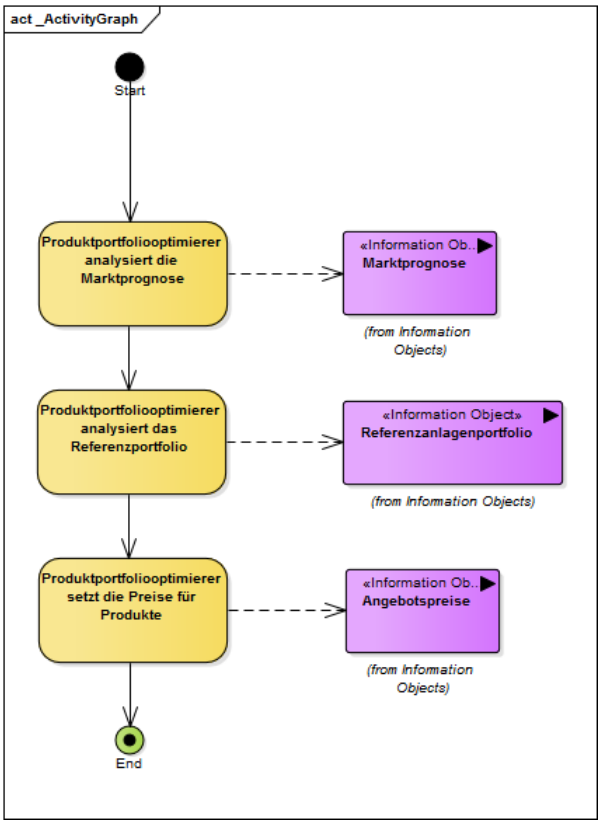


(b) Aktivitätsdiagramm

Abbildung 104: PUC *Marktprognosen abfragen*

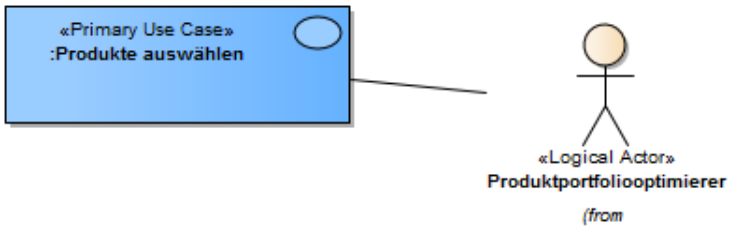


(a) Akteure

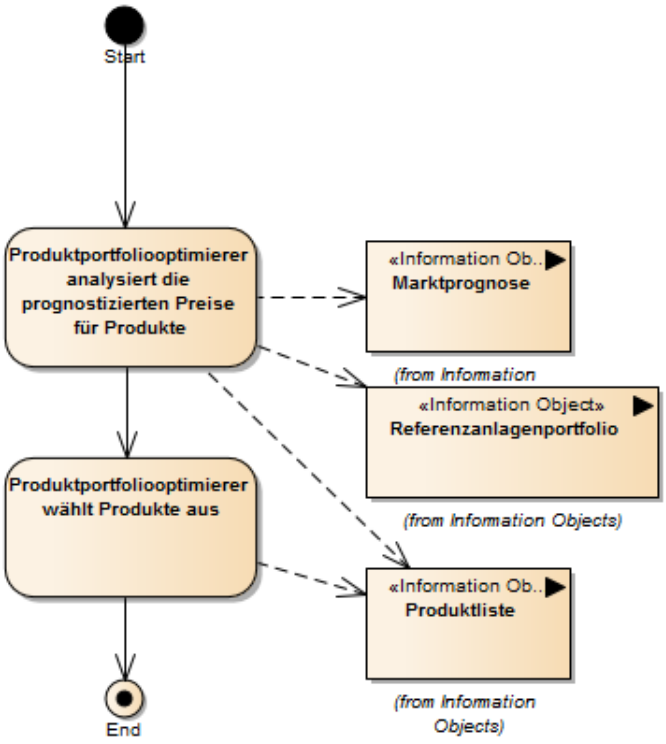


(b) Aktivitätsdiagramm

Abbildung 105: PUC *Preise festlegen*

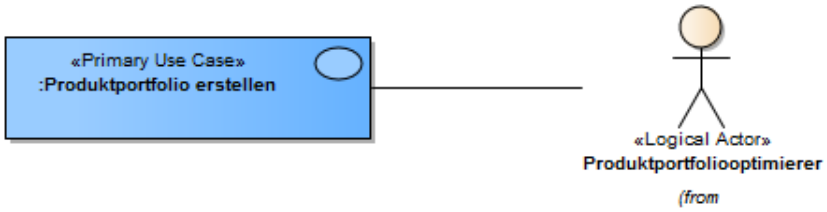


(a) Akteure

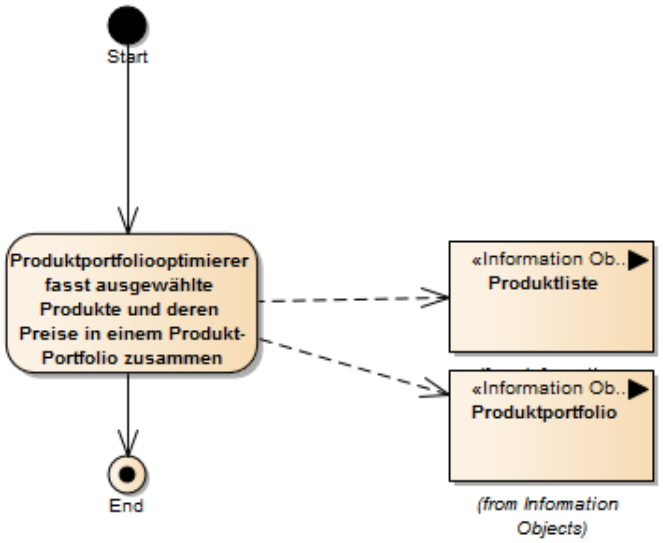


(b) Aktivitätsdiagramm

Abbildung 106: PUC *Produkte auswählen*

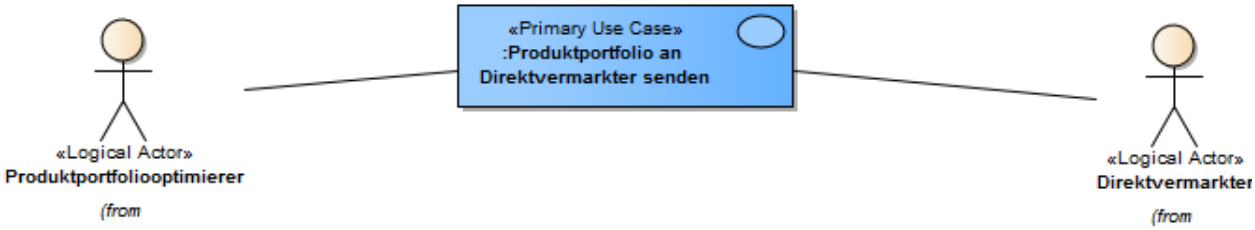


(a) Akteure

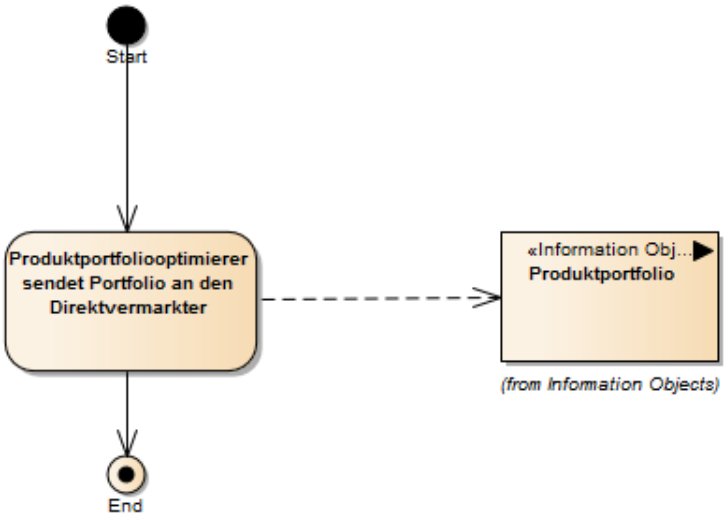


(b) Aktivitätsdiagramm

Abbildung 107: PUC *Produktportfolio erstellen*

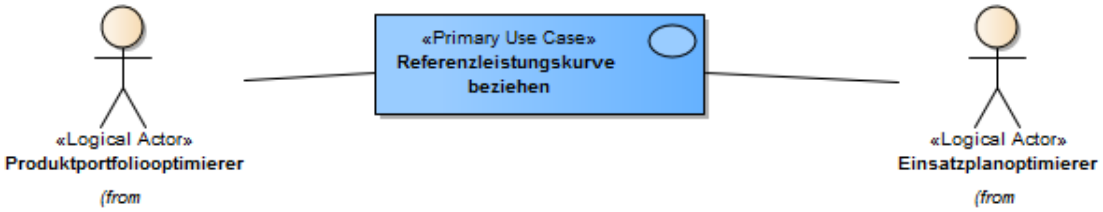


(a) Akteure

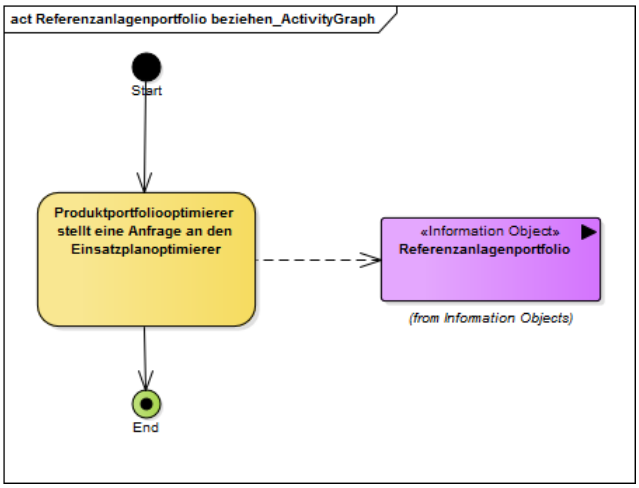


(b) Aktivitätsdiagramm

Abbildung 108: PUC *Produktportfolio an Direktvermarkter senden*

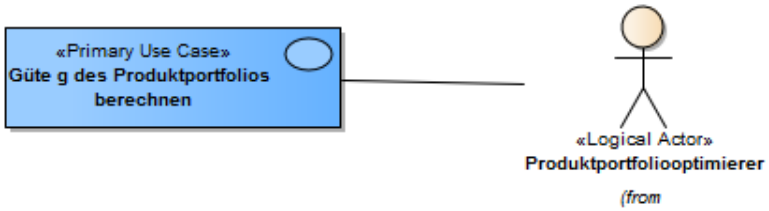


(a) Akteure

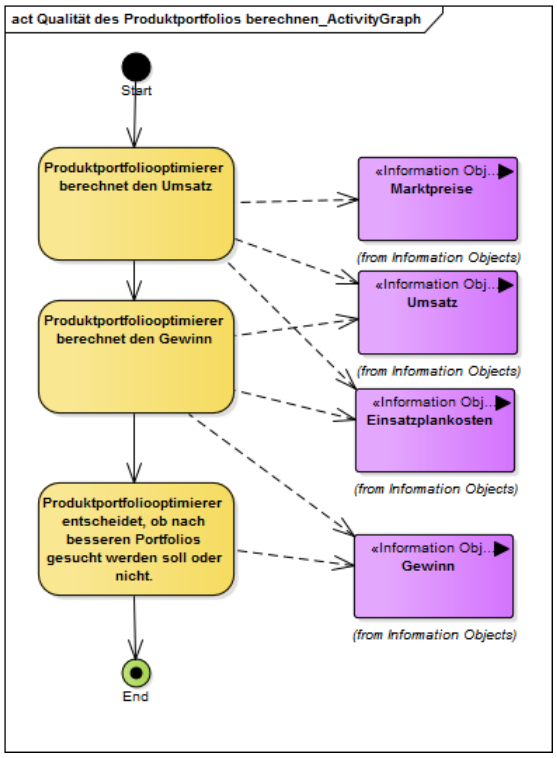


(b) Aktivitätsdiagramm

Abbildung 109: PUC *Referenzleistungskurve beziehen*



(a) Akteure



(b) Aktivitätsdiagramm

Abbildung 110: PUC *Güte g des Produktportfolios berechnen*

D.2. SGAM Information Layer

D.2.1. HLUC *VK initialisieren*

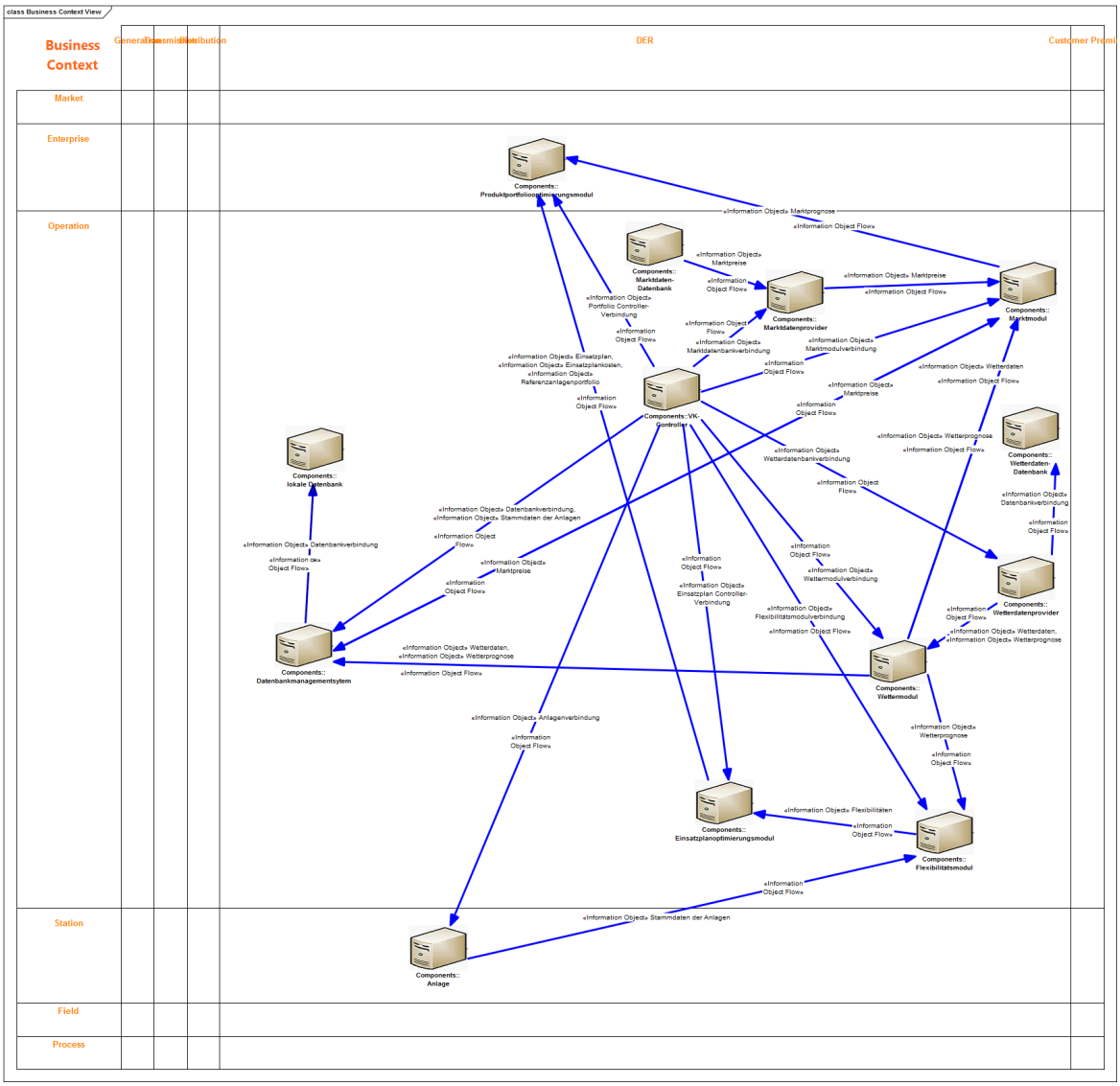


Abbildung 111: Business Context View für HLUC *Virtuelles Kraftwerk initialisieren*

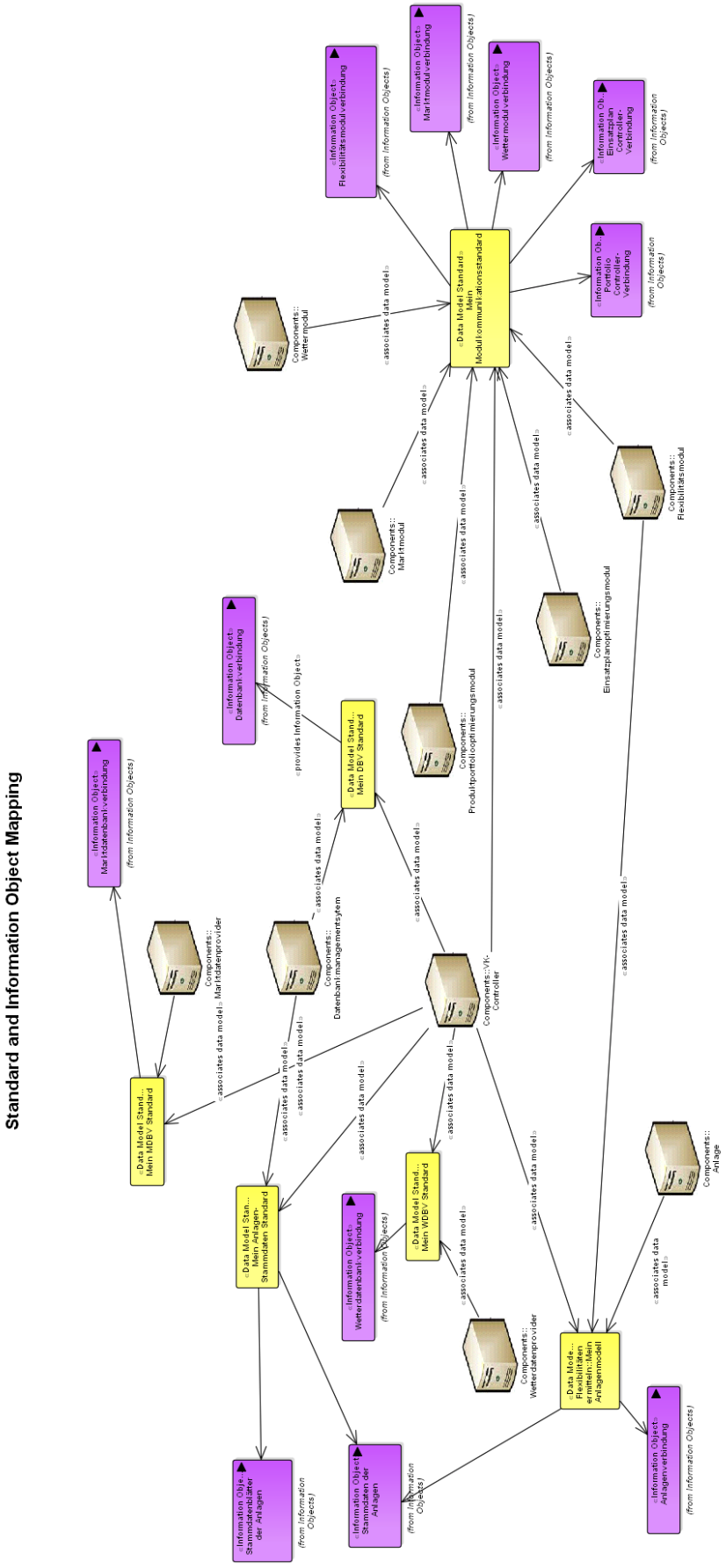


Abbildung 112: Standard and Information Object Mapping für HLUC *Virtuelles Kraftwerk initialisieren*

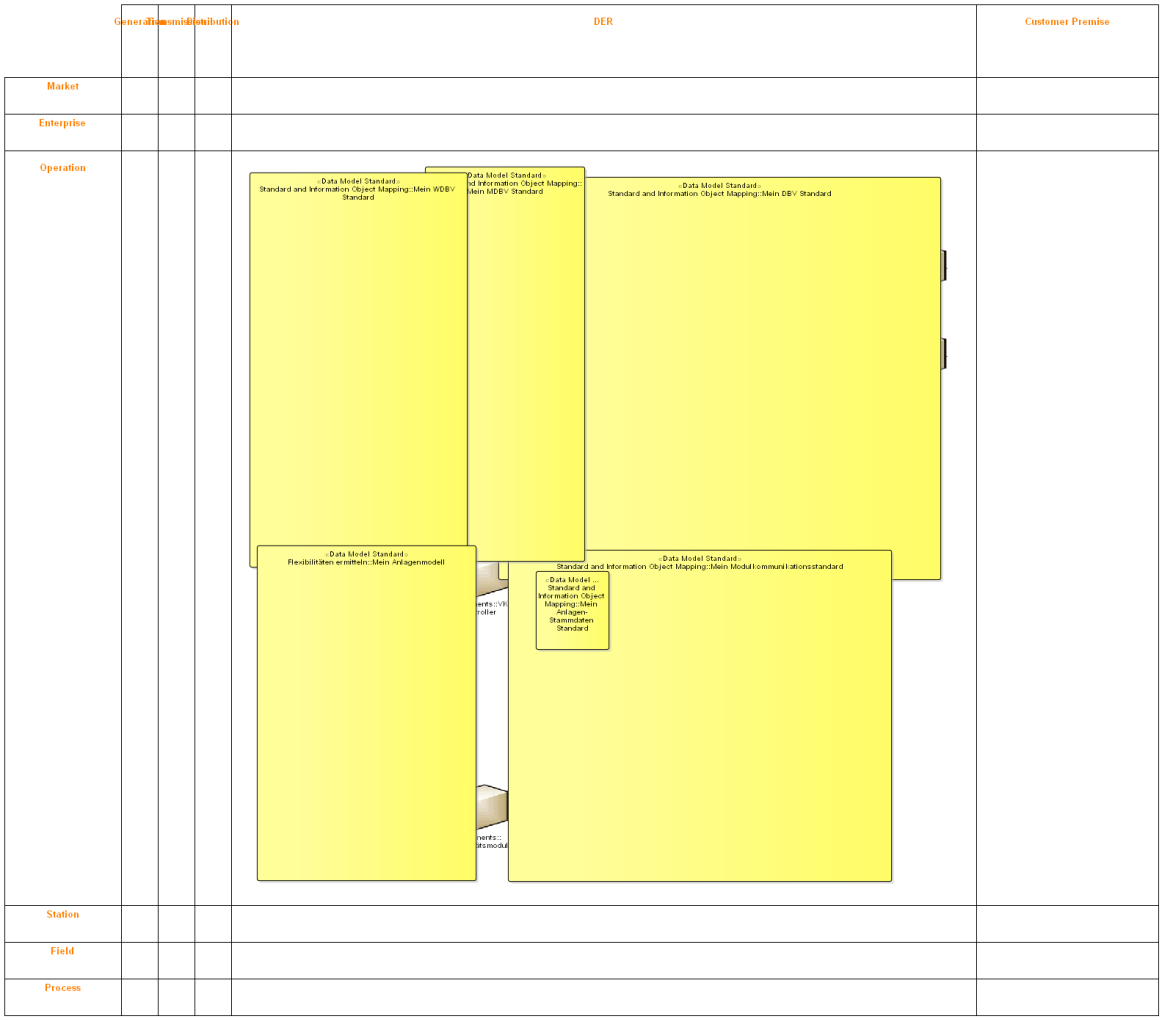


Abbildung 113: Canonical Data Model View für HLUC *Virtuelles Kraftwerk initialisieren*

D.2.2. HLUC *Marktprognose erstellen*

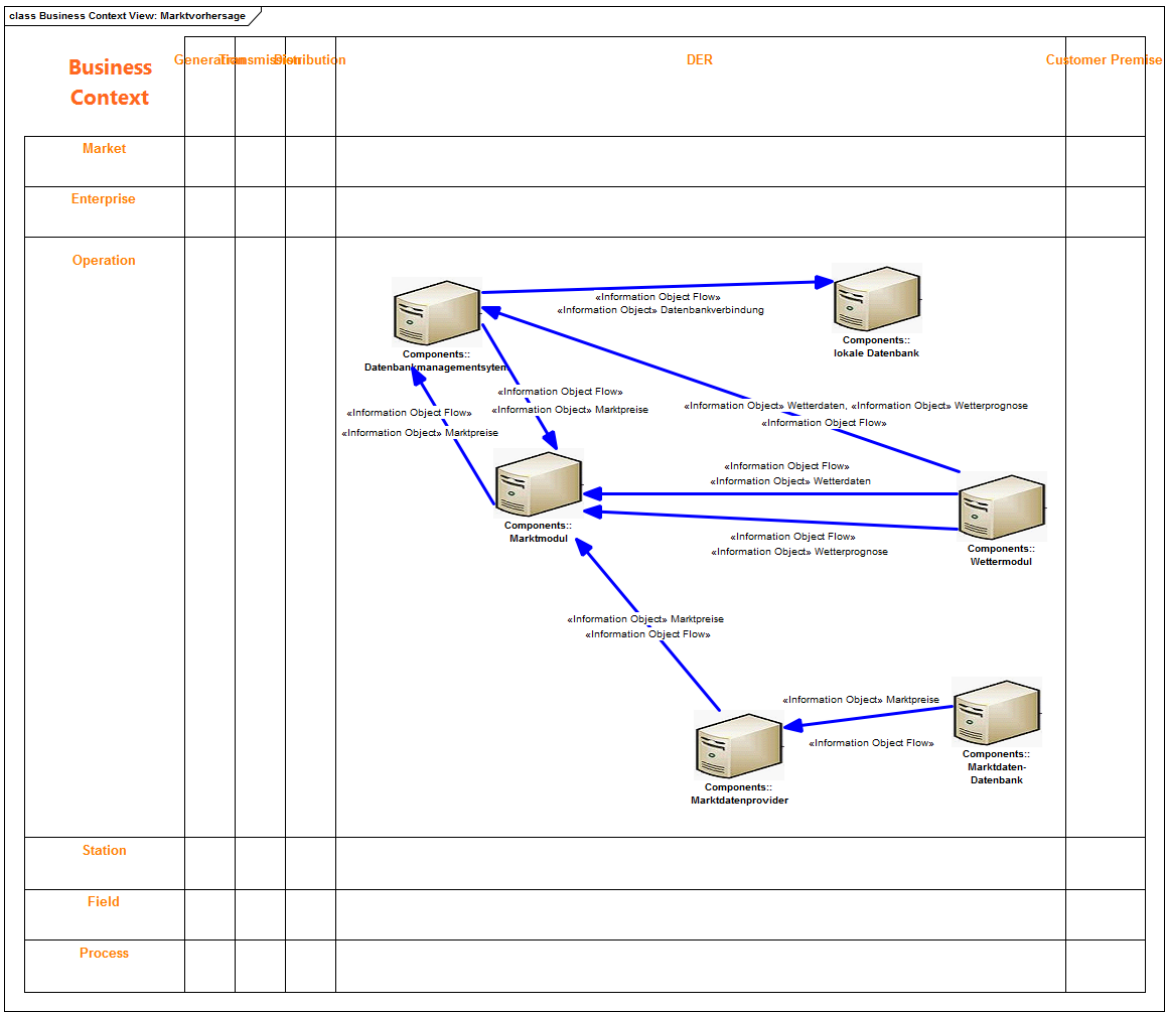


Abbildung 114: Business Context View für HLUC *Marktvorhersage*

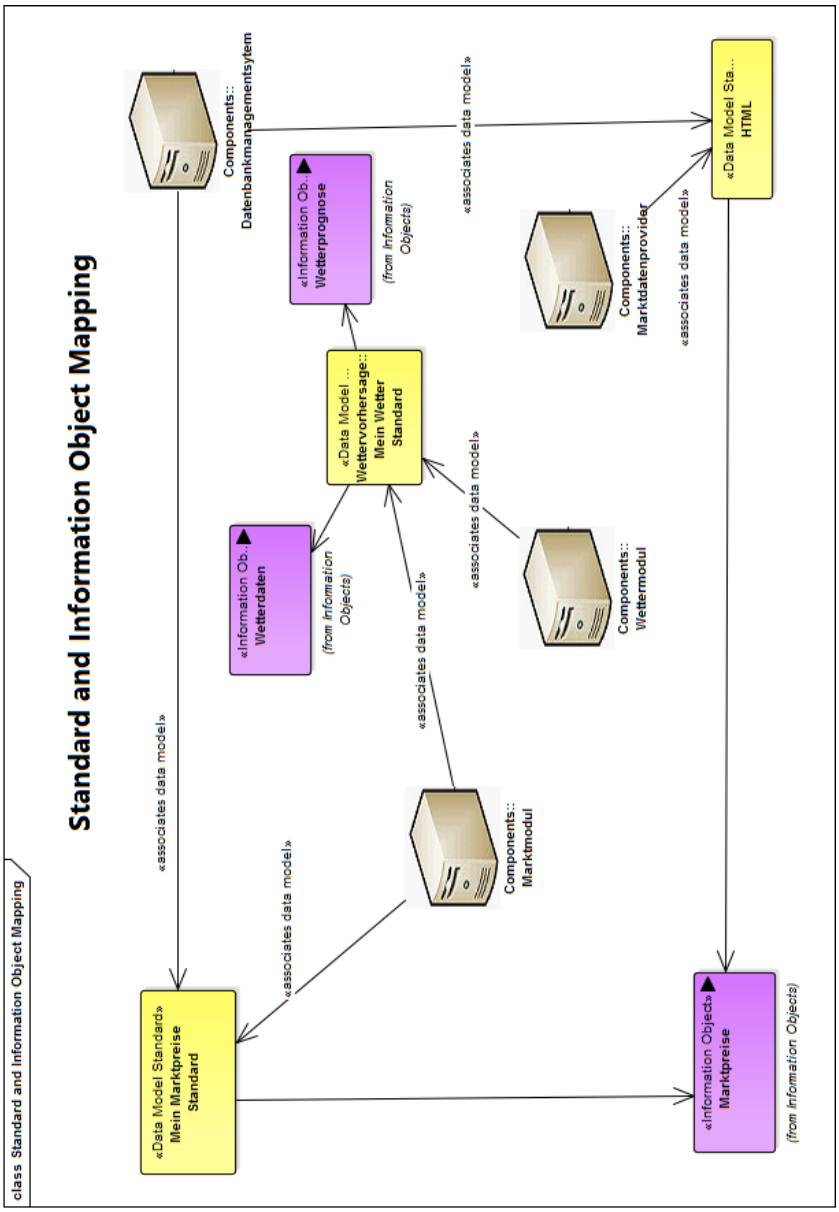


Abbildung 115: Standard and Information Object Mapping für HLUC *Marktvorhersage*

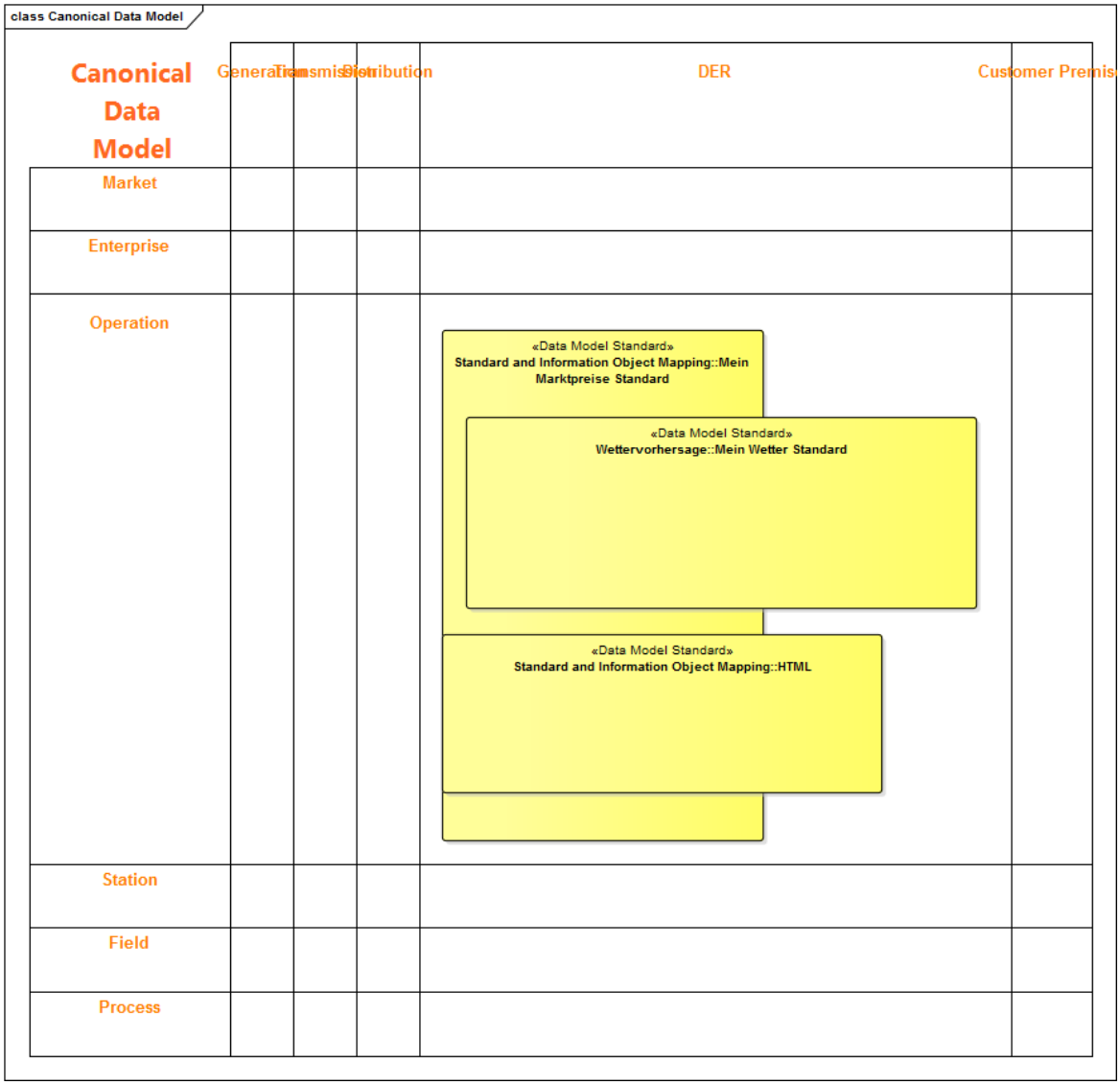


Abbildung 116: Canonical Data Model View für HLUC *Marktvorhersage*

D.2.3. HLUC *Flexibilitäten ermitteln*

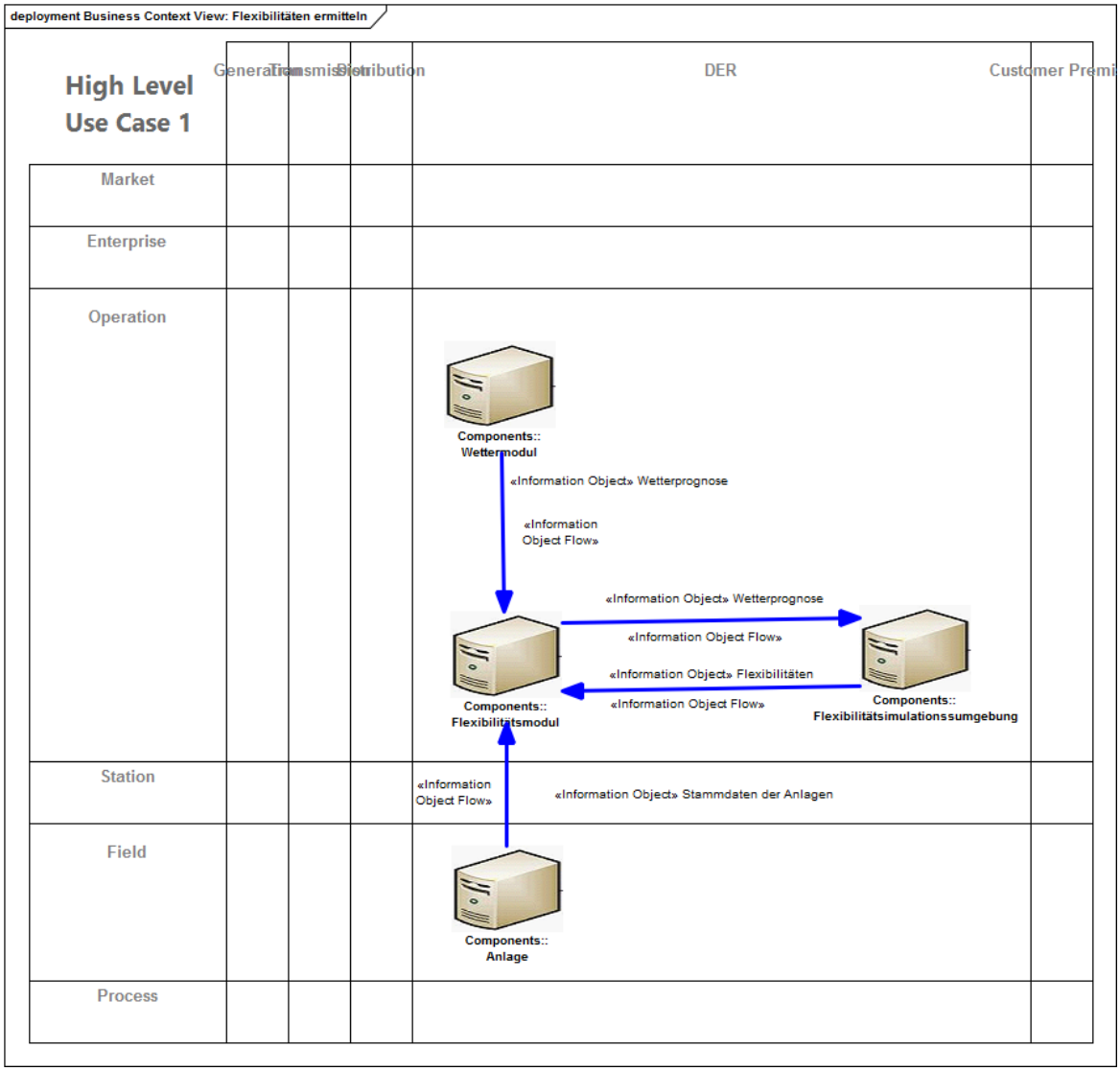


Abbildung 117: Business Context View für HLUC *Flexibilitäten ermitteln*

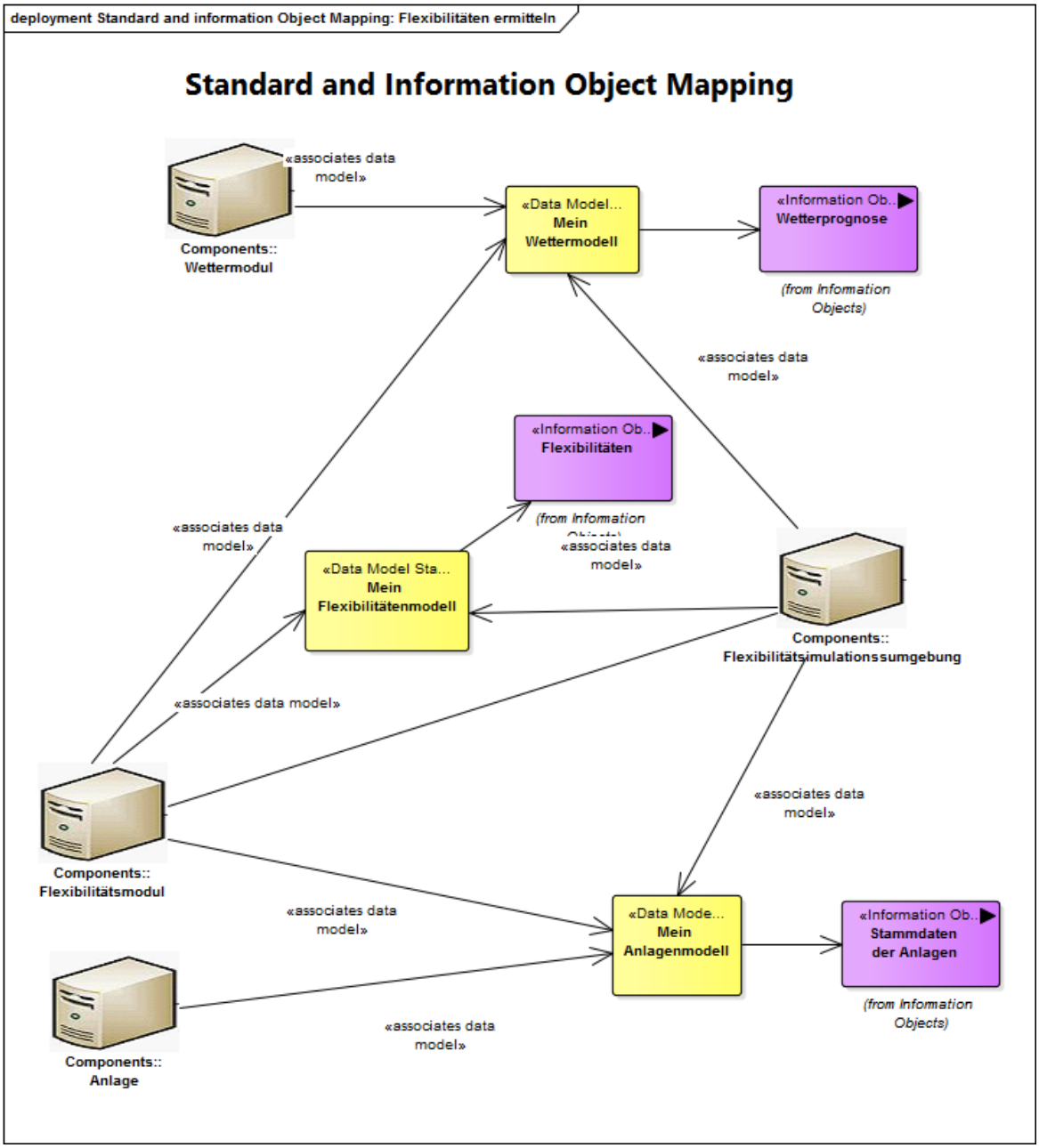


Abbildung 118: Standard and Information Object Mapping für HLUC *Flexibilitäten ermitteln*

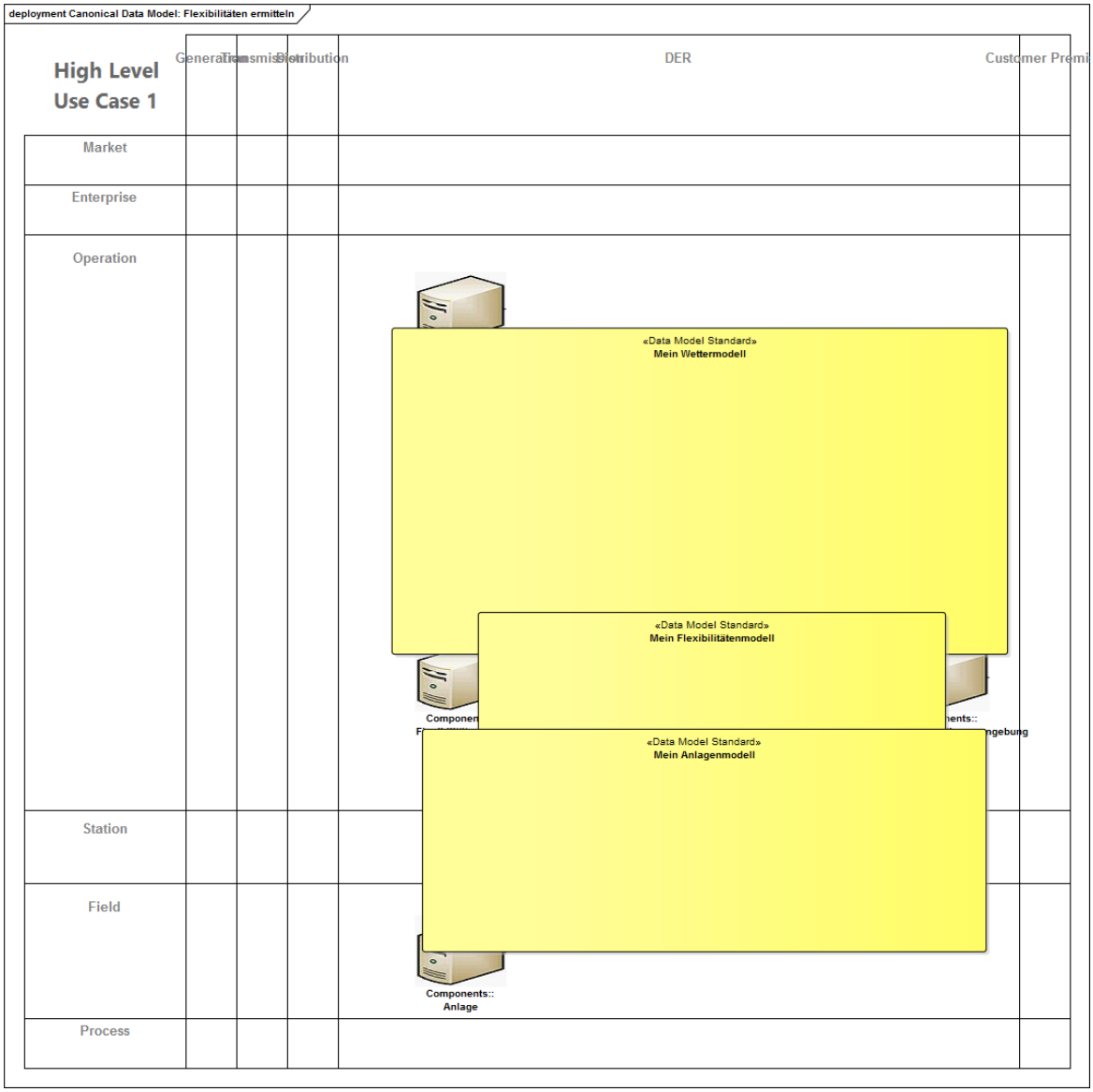


Abbildung 119: Canonical Data Model View für HLUC *Flexibilitäten ermitteln*

D.2.4. HLUC *Einsatzplanoptimierung*

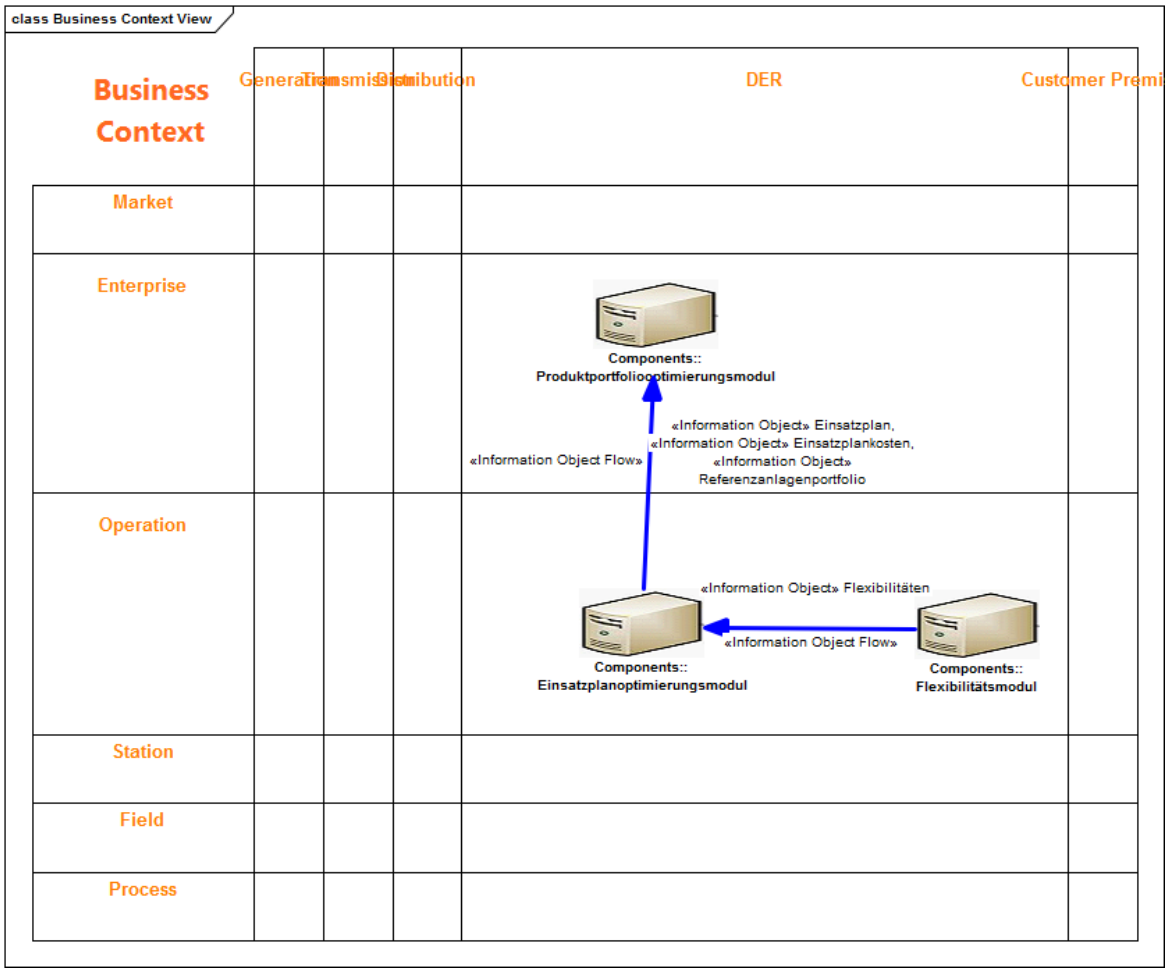


Abbildung 120: Business Context View für HLUC *Einsatzplanoptimierung*

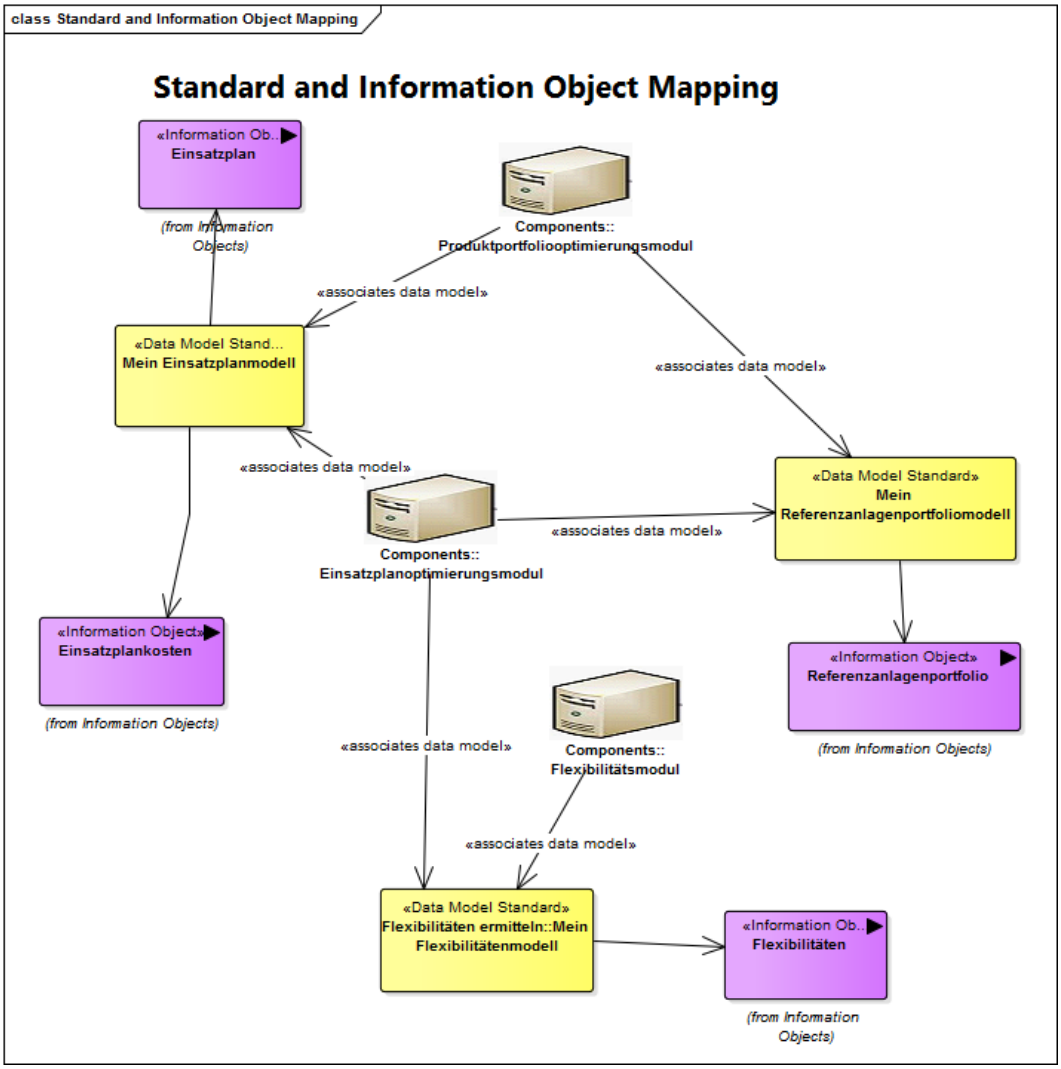


Abbildung 121: Standard and Information Object Mapping für HLUC *Einsatzplanoptimierung*

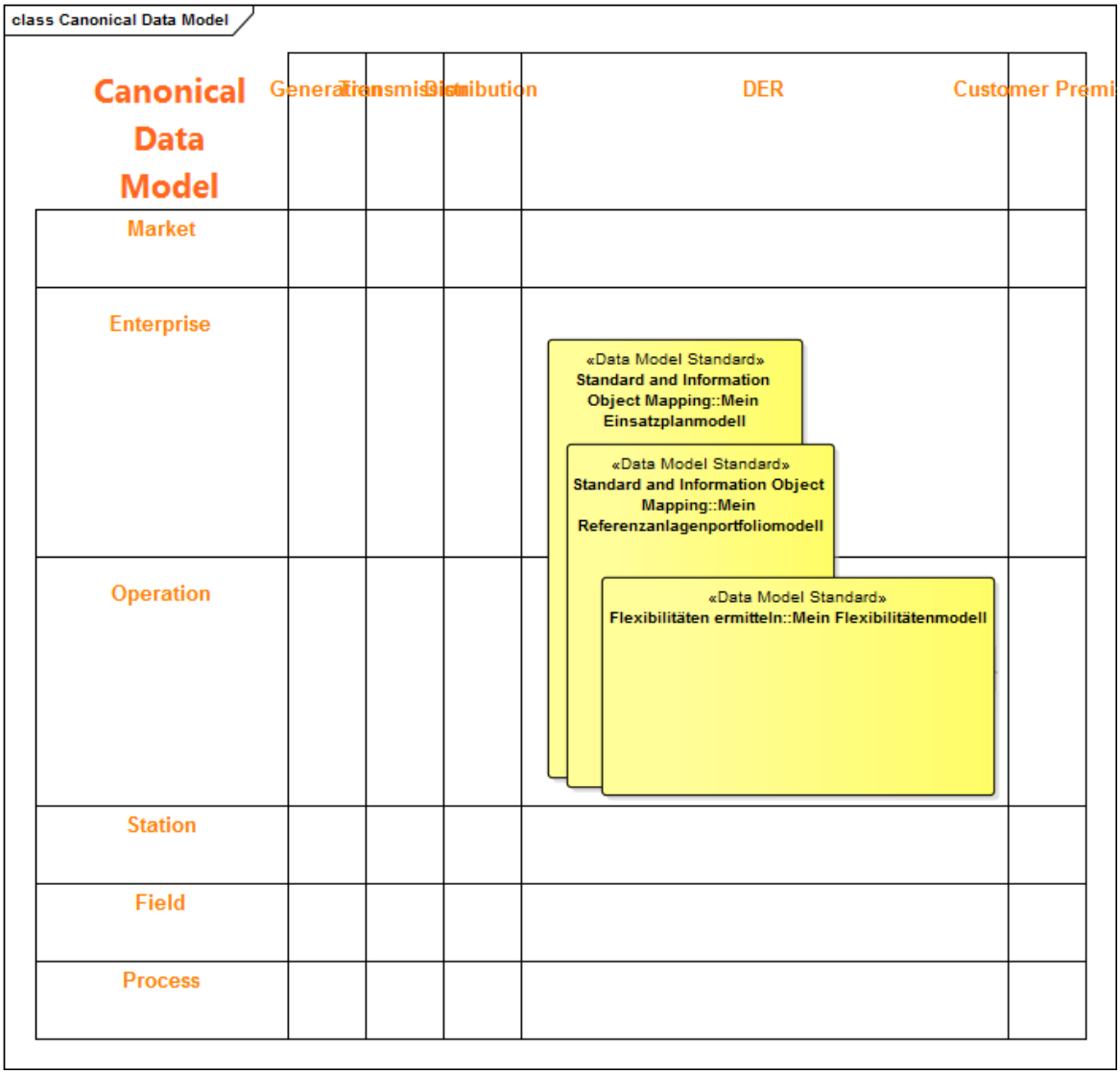


Abbildung 122: Canonical Data Model View für HLUC *Einsatzplanoptimierung*

D.2.5. HLUC *Produktportfoliooptimierung*

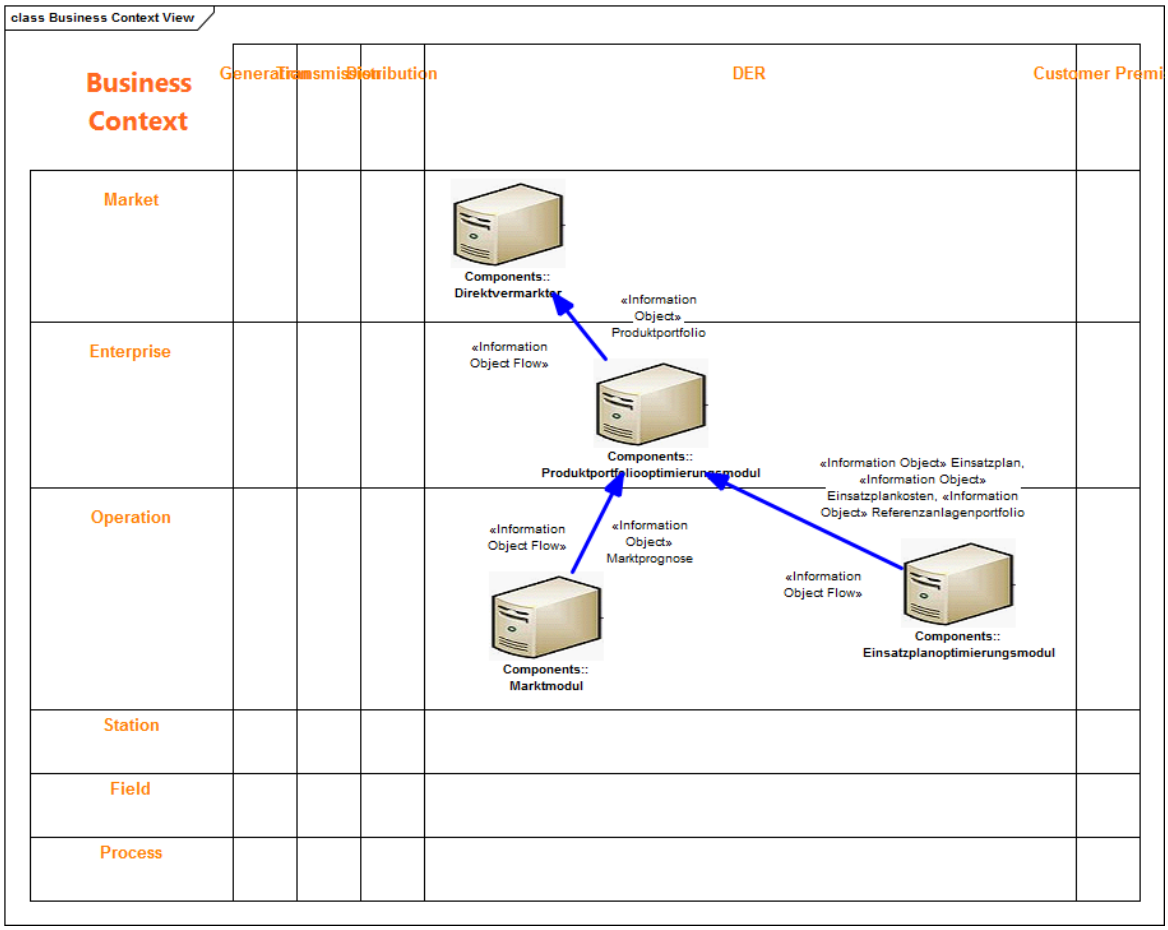


Abbildung 123: Business Context View für HLUC *Produktportfoliooptimierer*

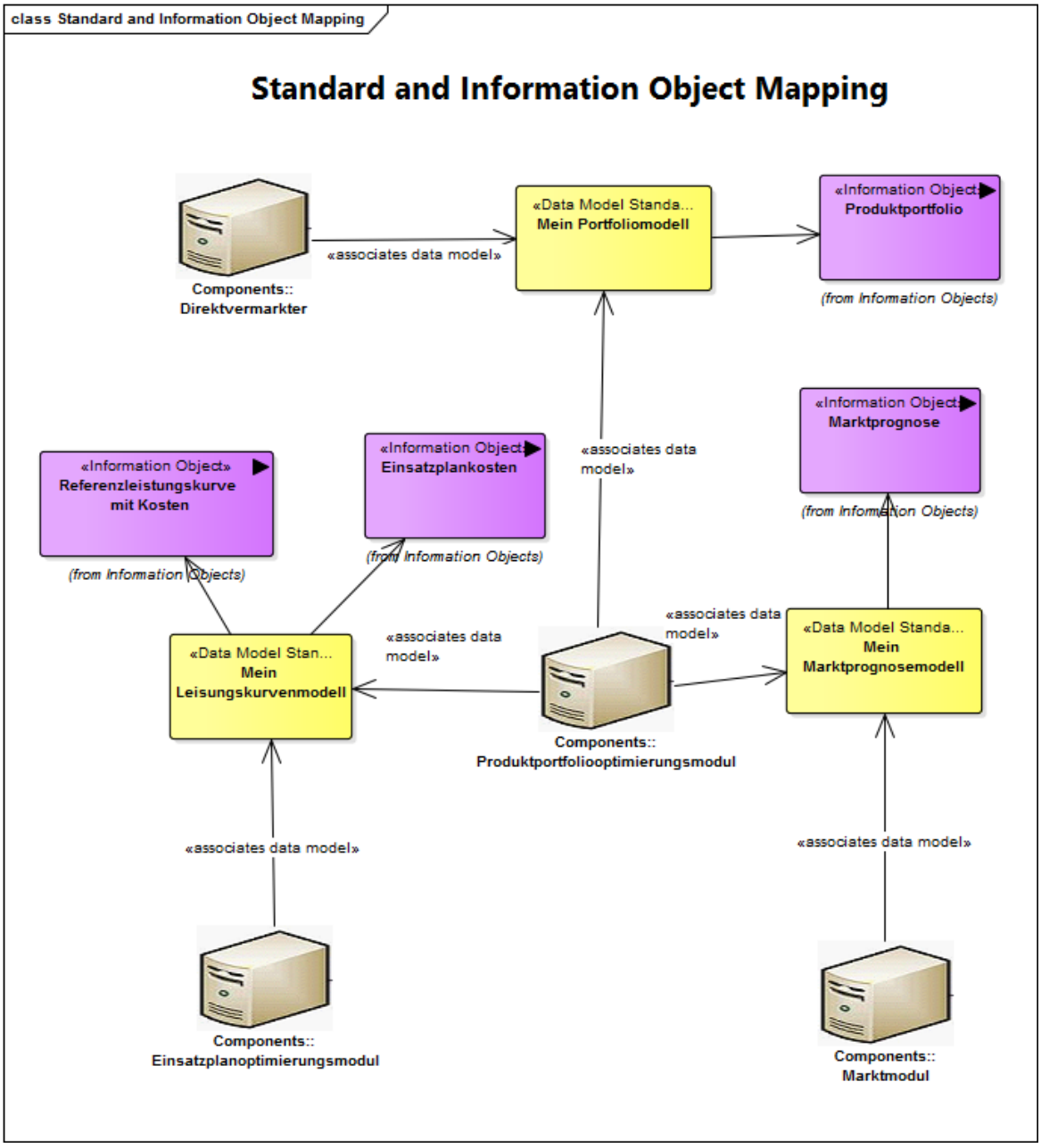


Abbildung 124: Standard and Information Object Mapping für HLUC *Produktportfoliooptimierer*

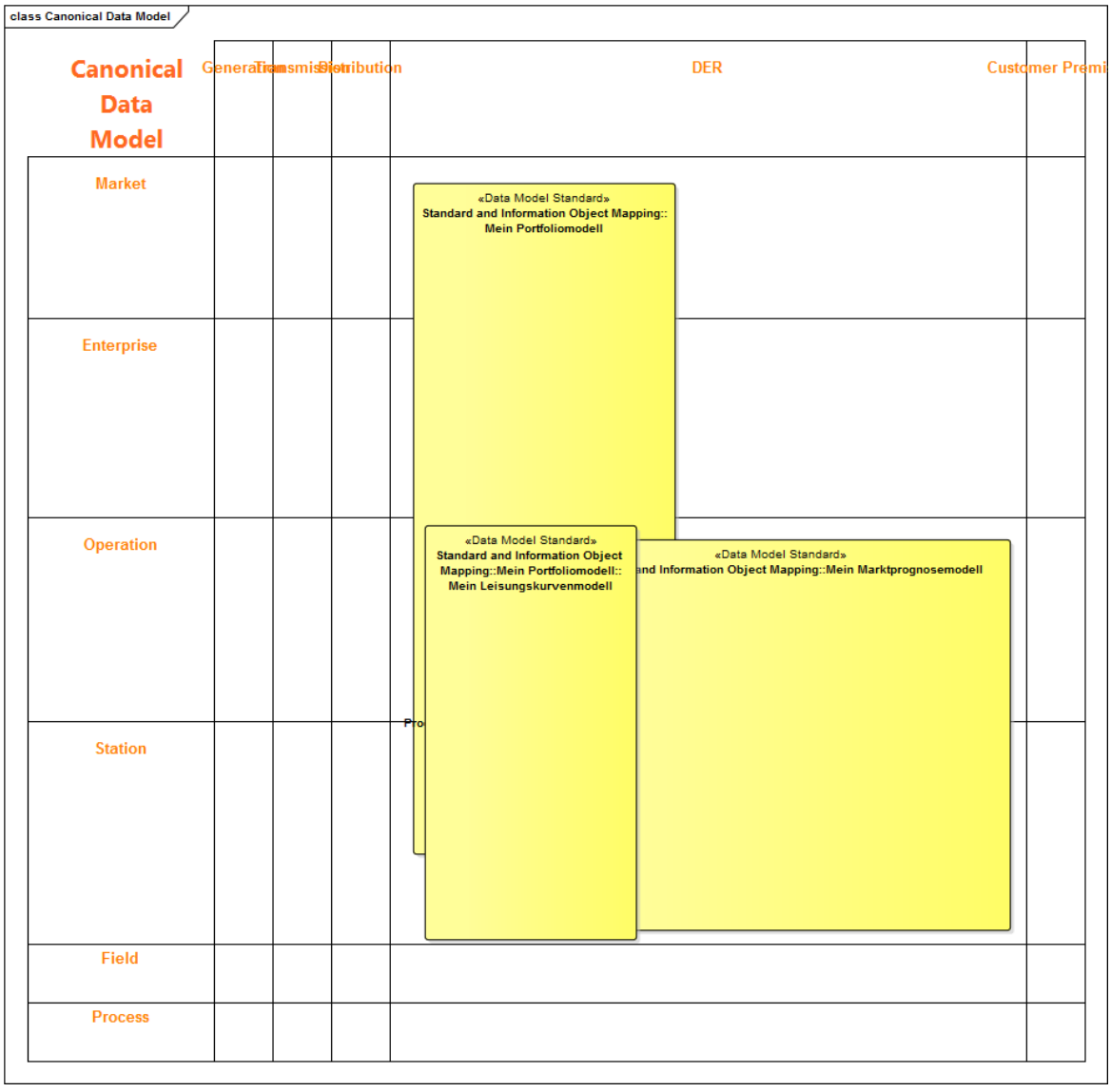


Abbildung 125: Canonical Data Model View für HLUC *Produktportfoliooptimierer*

D.3. SGAM Component Layer

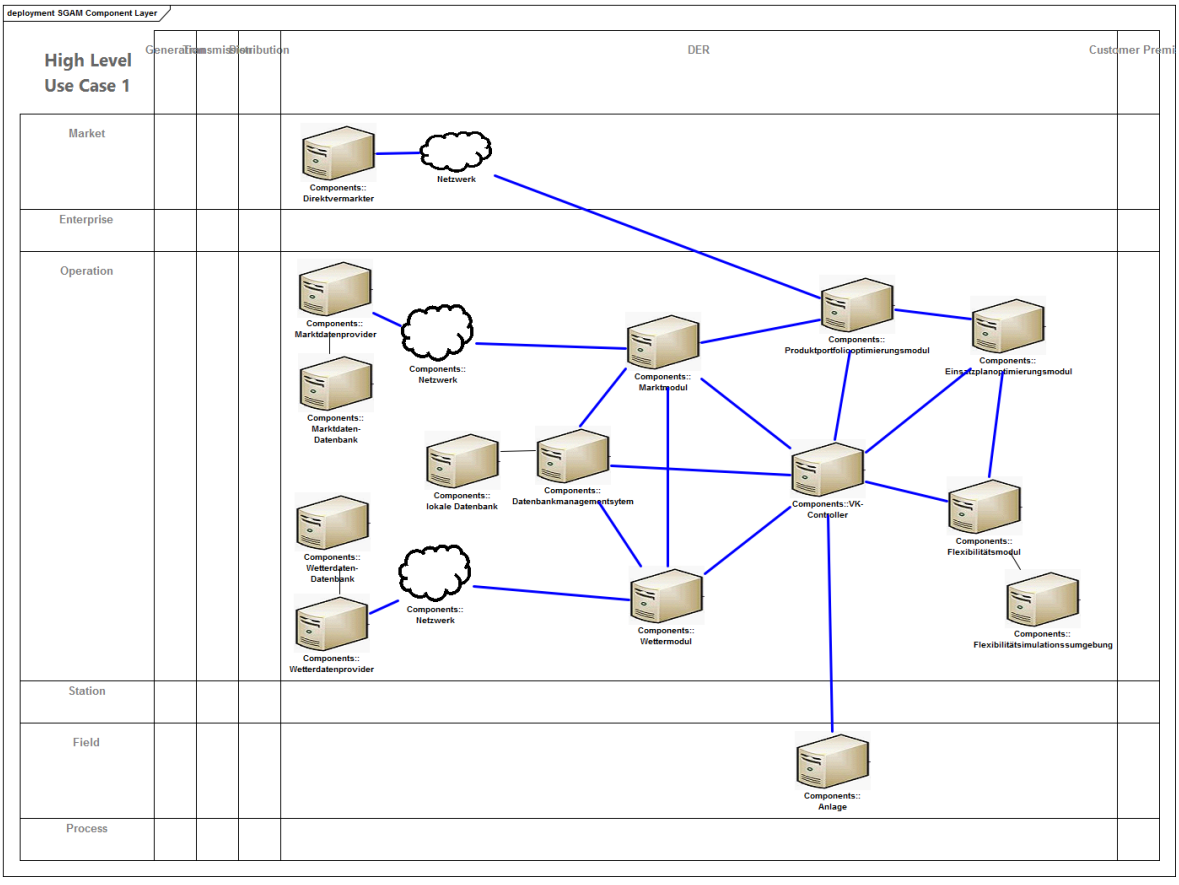


Abbildung 126: Component Layer Übersicht

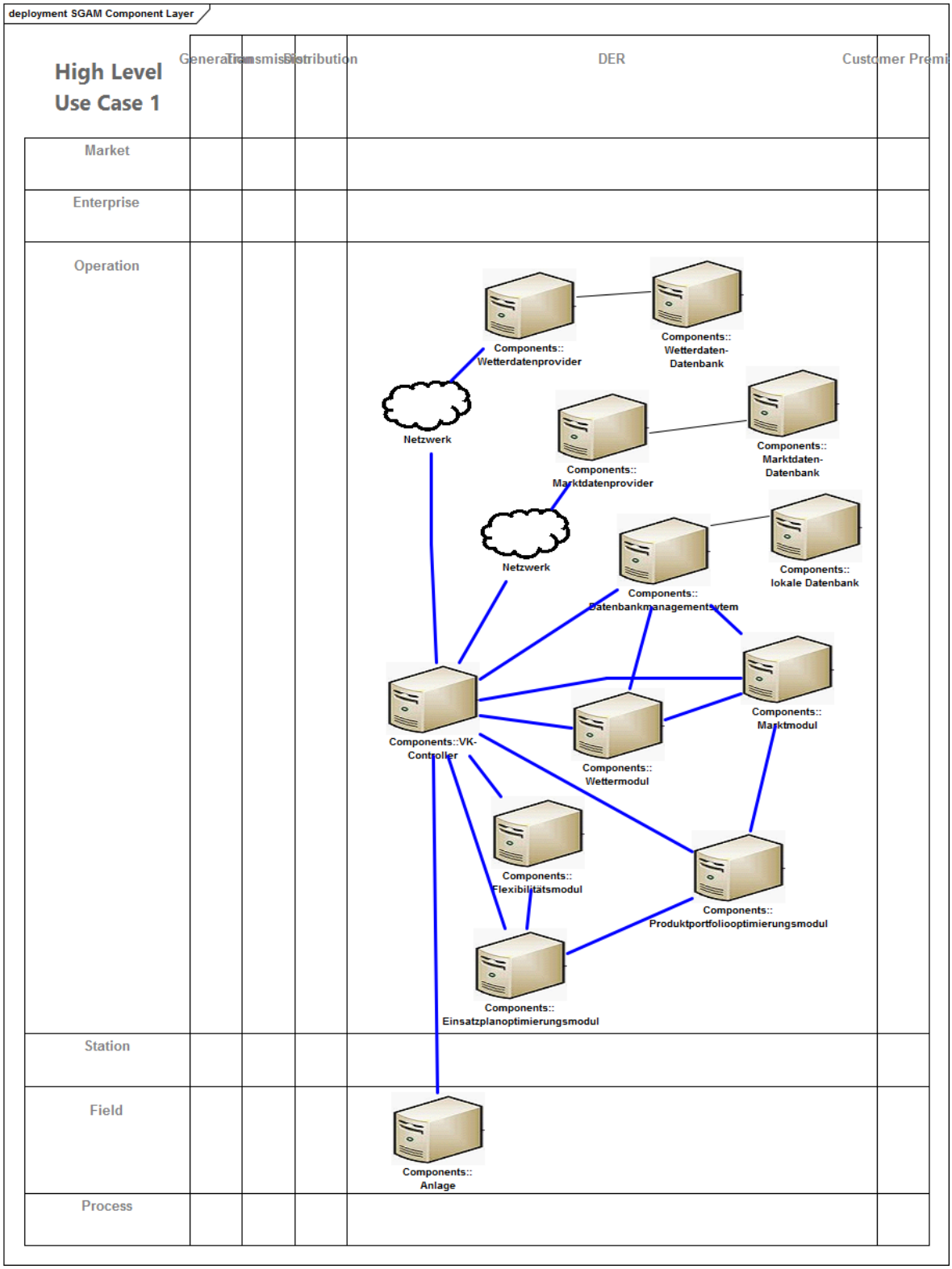


Abbildung 127: Component Layer HLUC *VK initialisieren*

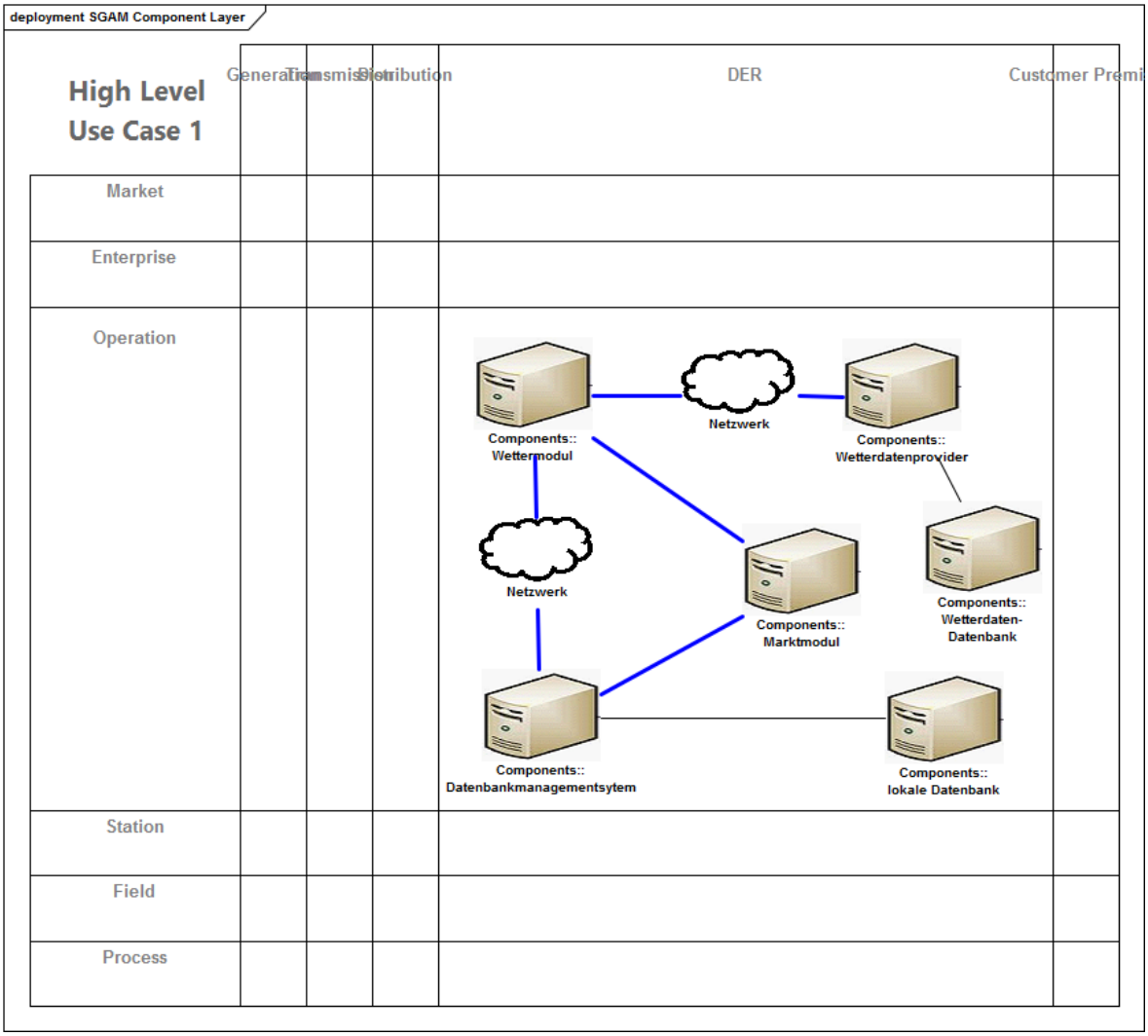


Abbildung 128: Component Layer HLUC *Wettervorhersage*

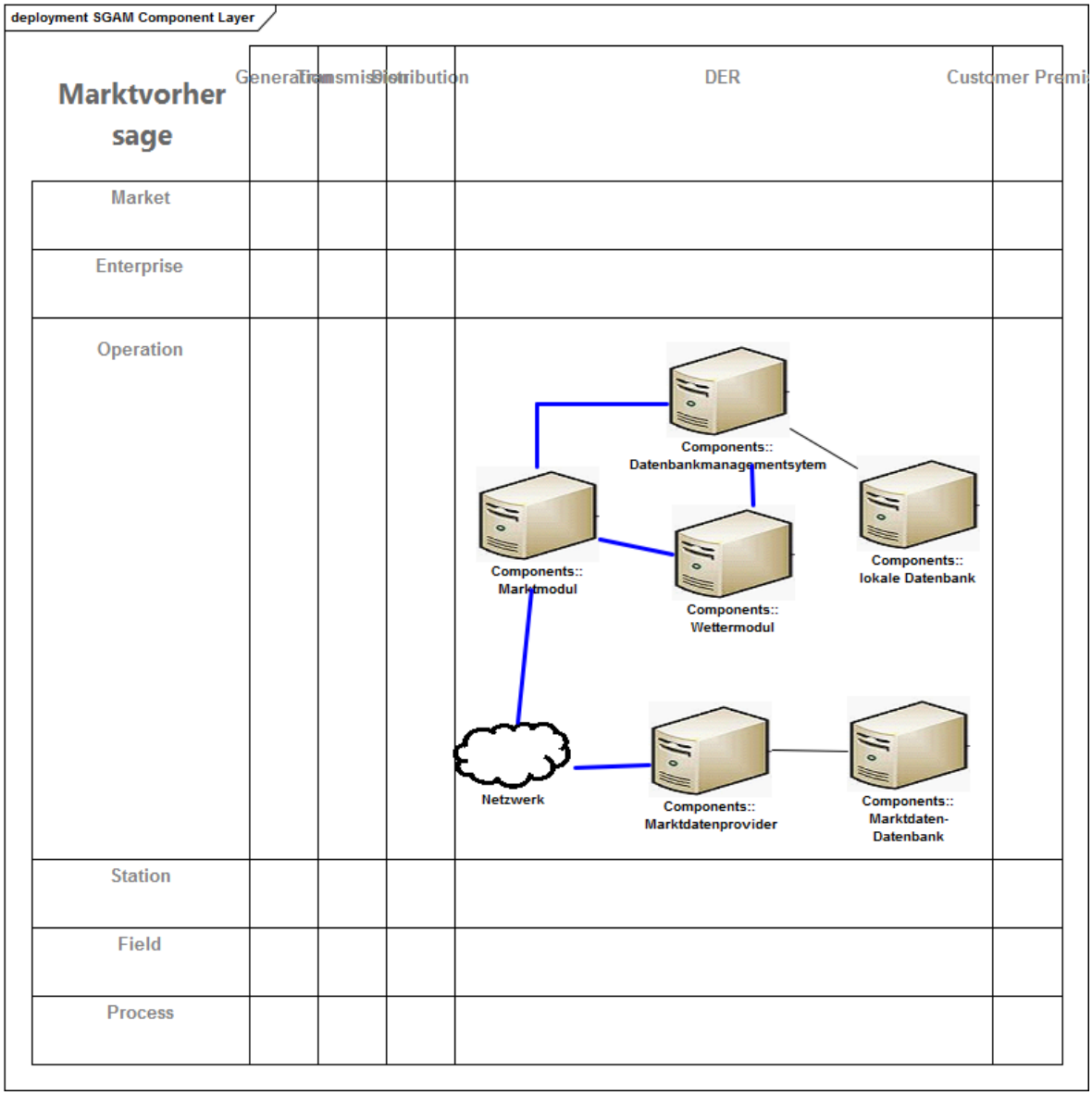


Abbildung 129: Component Layer HLUC *Marktvorhersage*

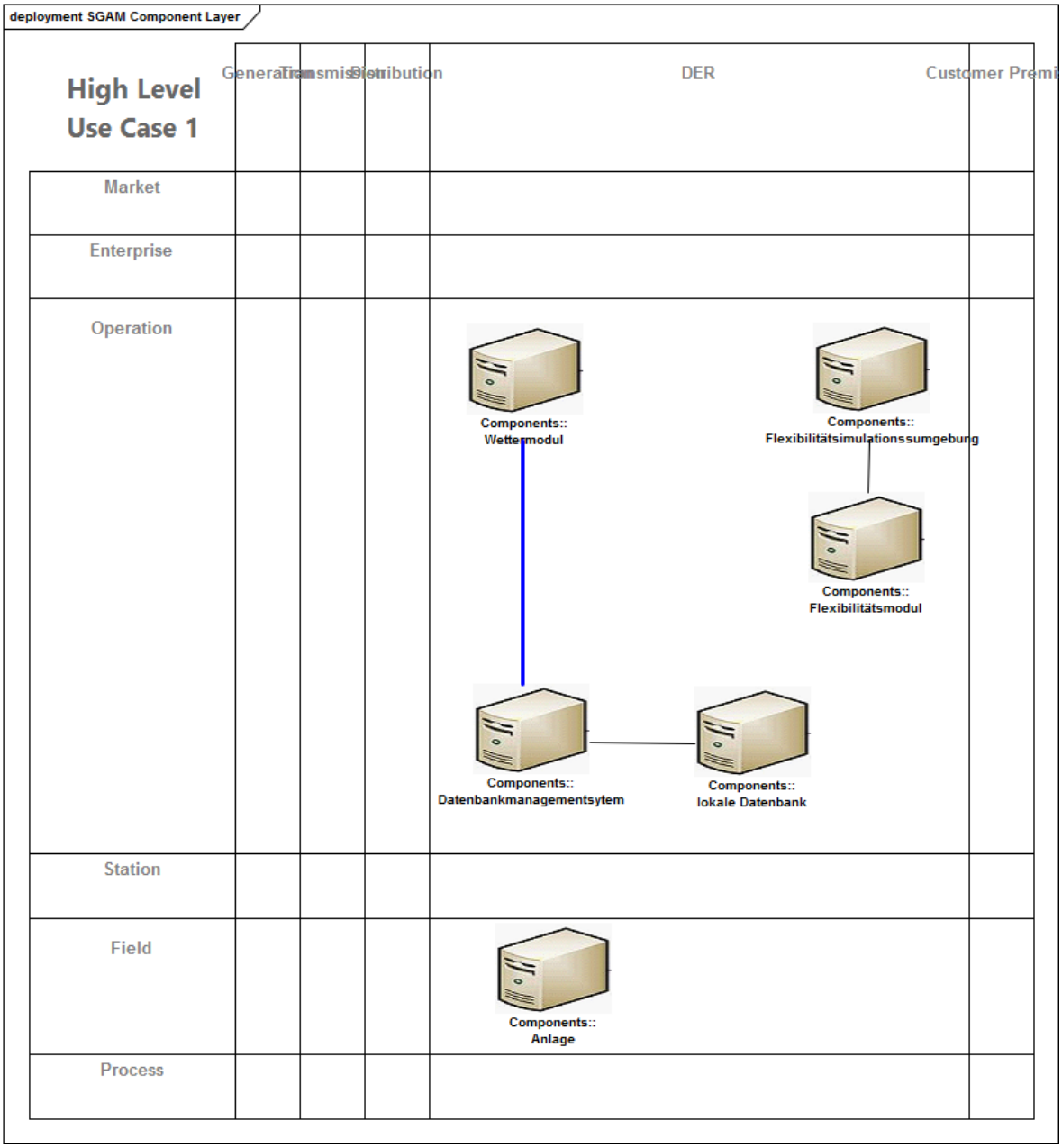


Abbildung 130: Component Layer HLUC *Flexibilitäten ermitteln*

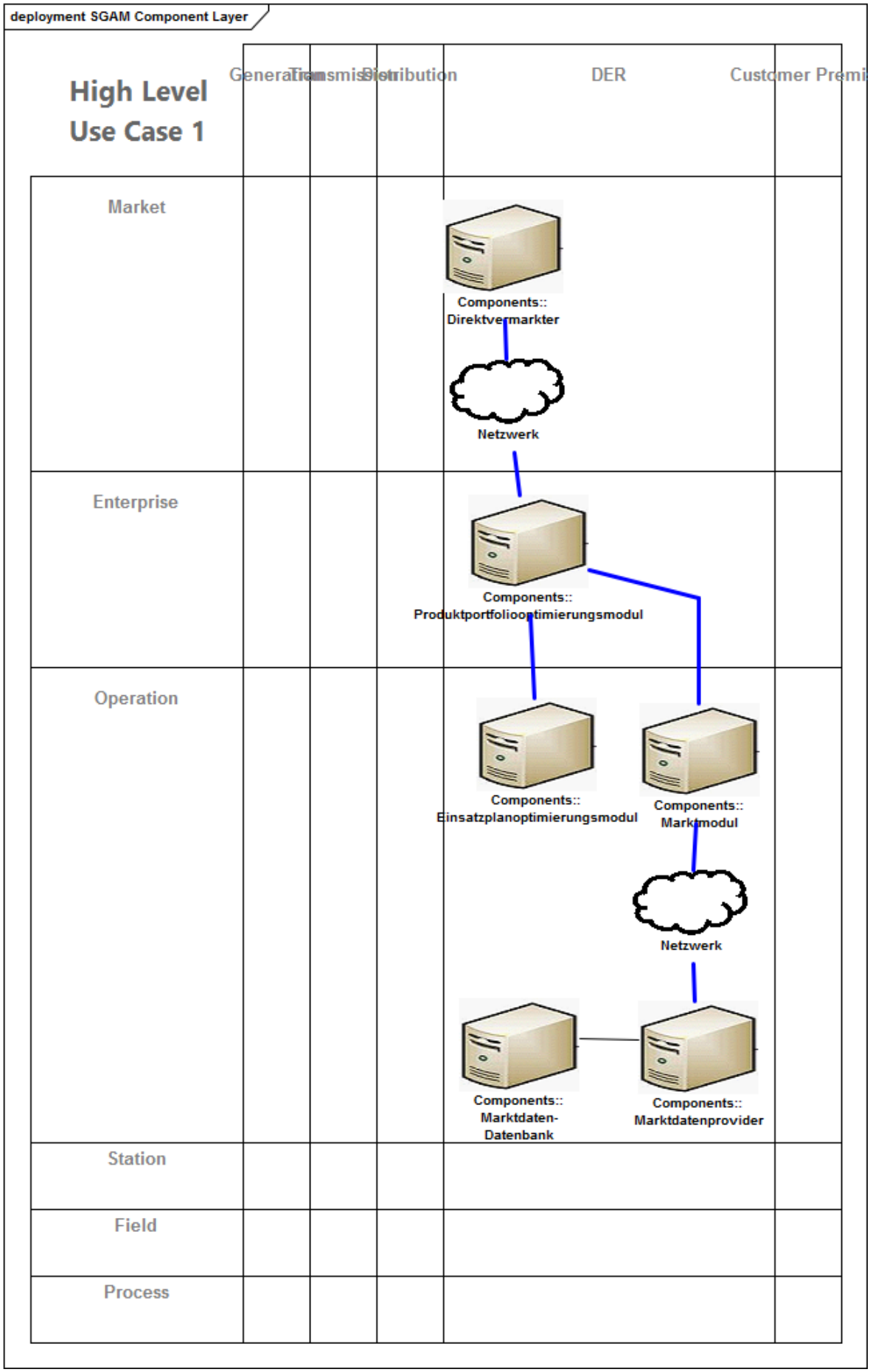


Abbildung 131: Component Layer HLUC *Produktportfoliooptimierung*

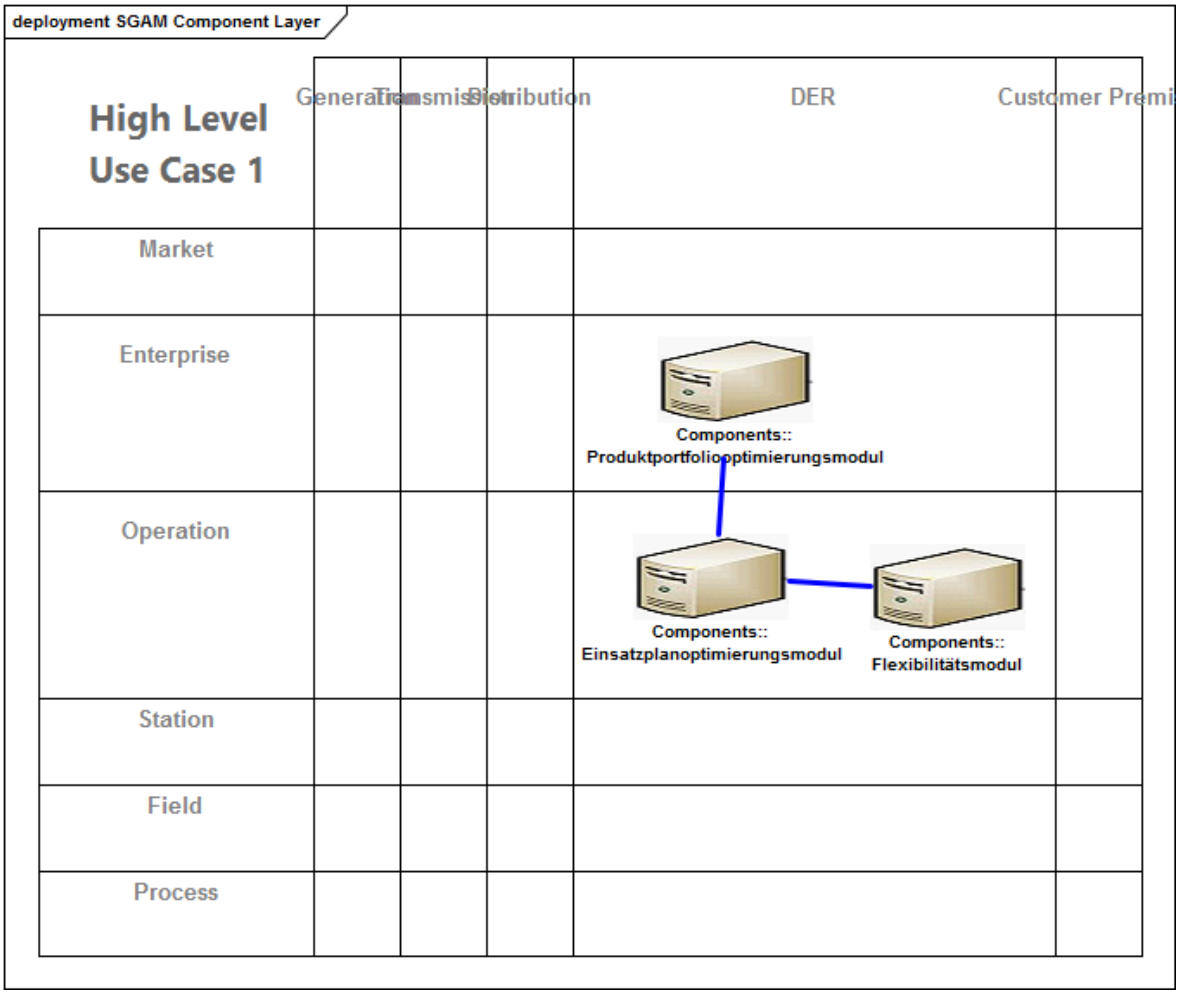


Abbildung 132: Component Layer HLUC *Einsatzplanoptimierung*

D.4. SGAM Communication Layer

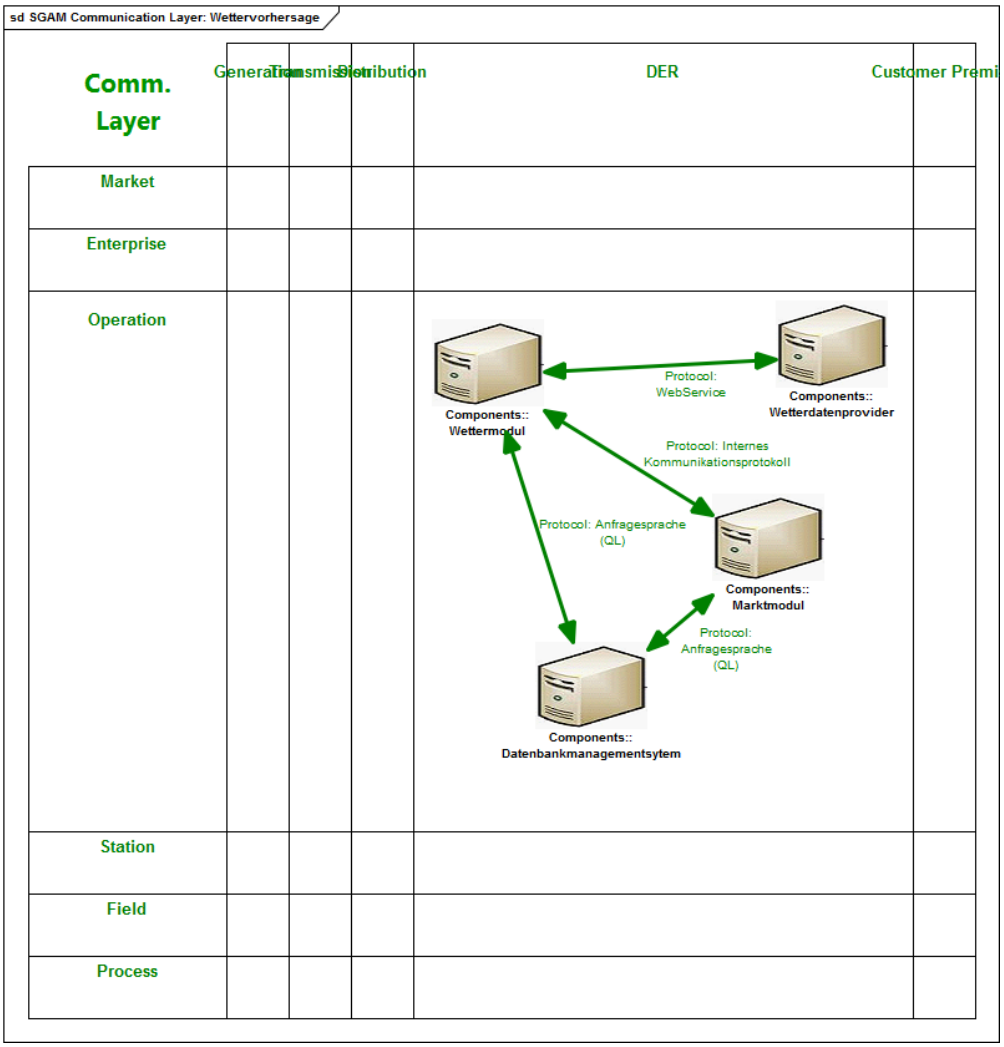


Abbildung 133: Communication Layer HLUC *Wettervorhersage*

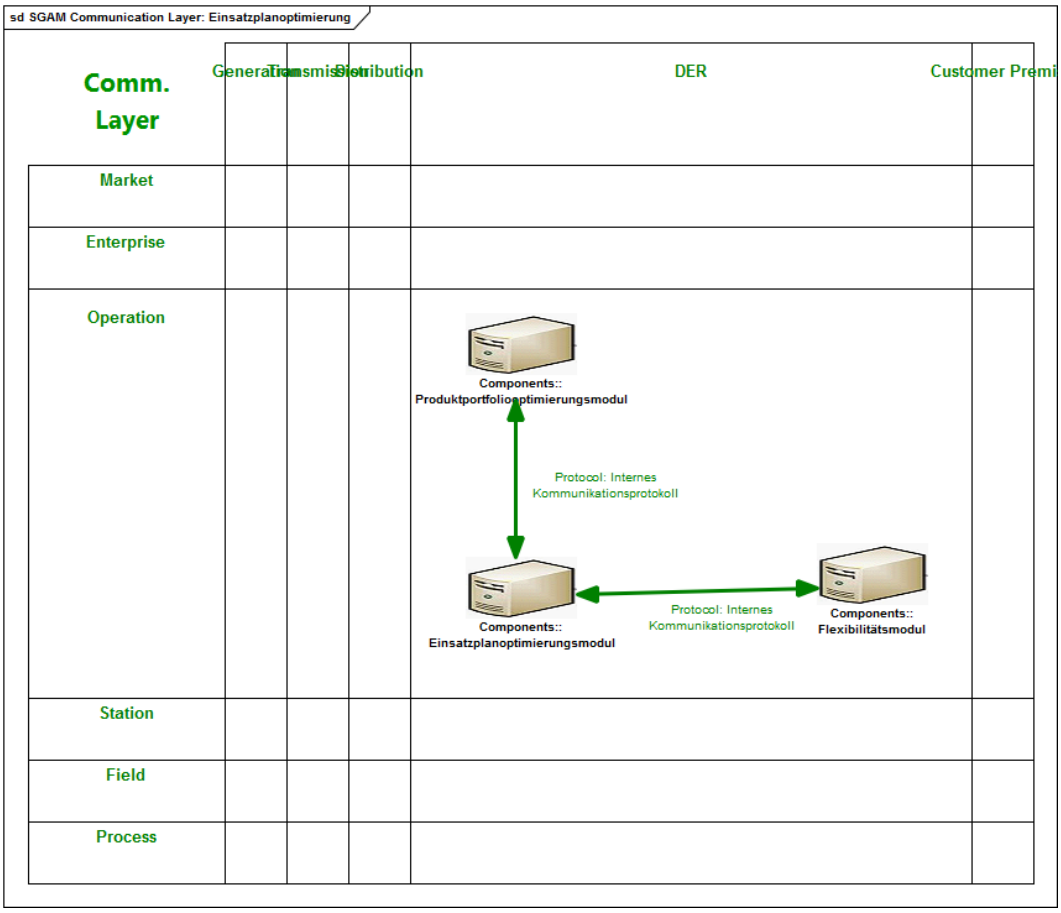


Abbildung 134: Communication Layer HLUC *Einsatzplanoptimierung*

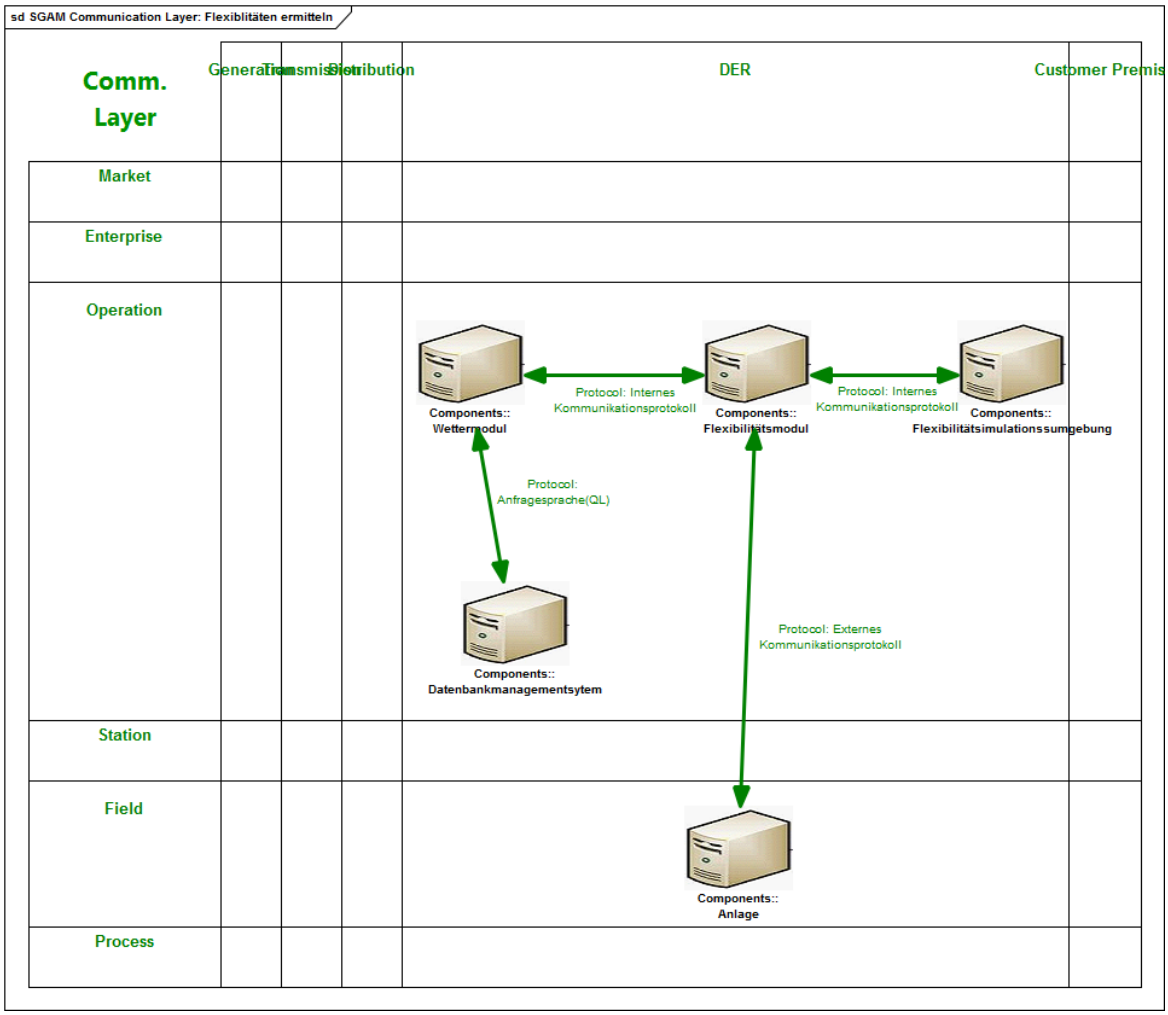


Abbildung 135: Communication Layer HLUC *Flexibilitäten ermitteln*

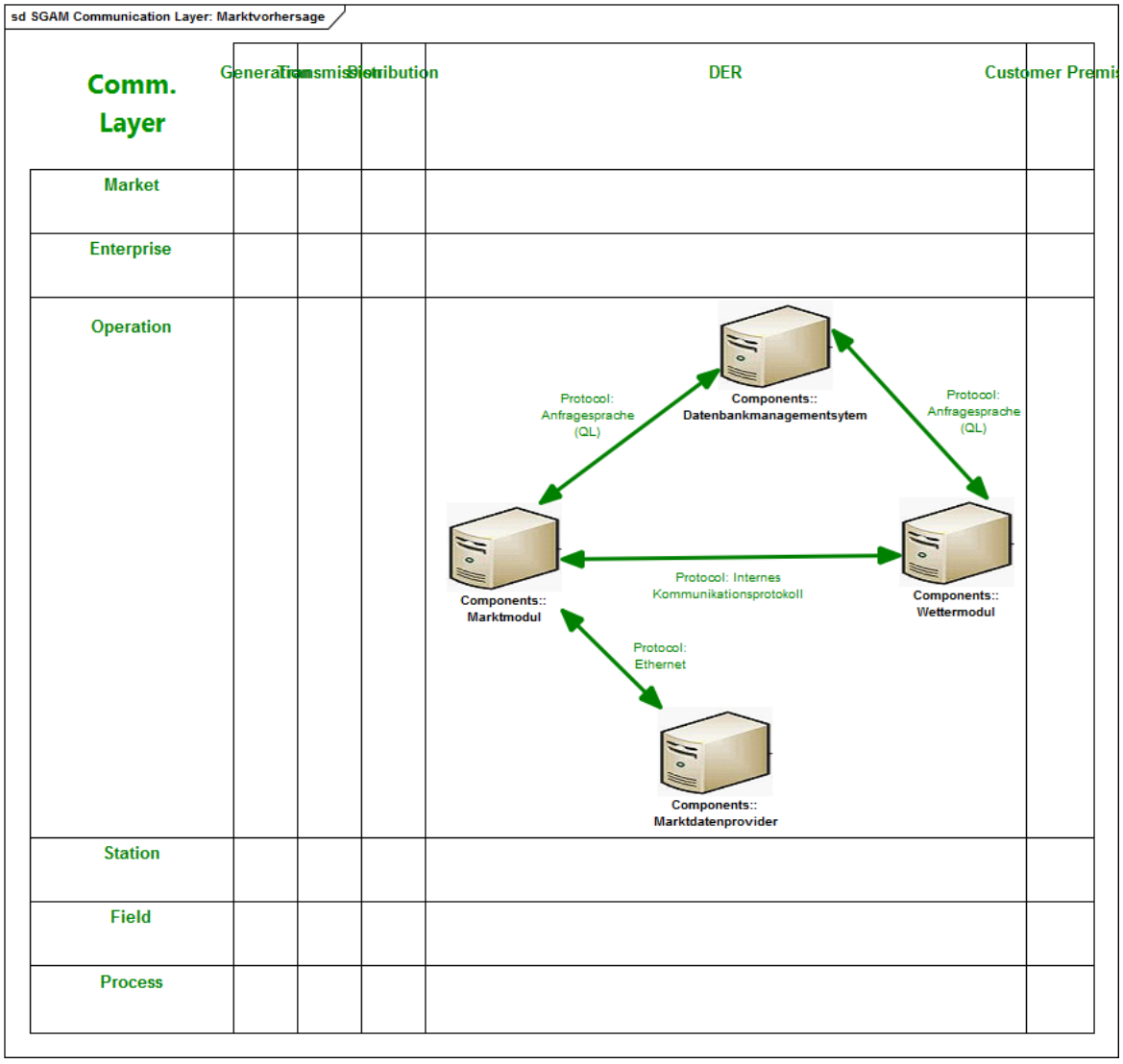


Abbildung 136: Communication Layer HLUC *Marktvorhersage*

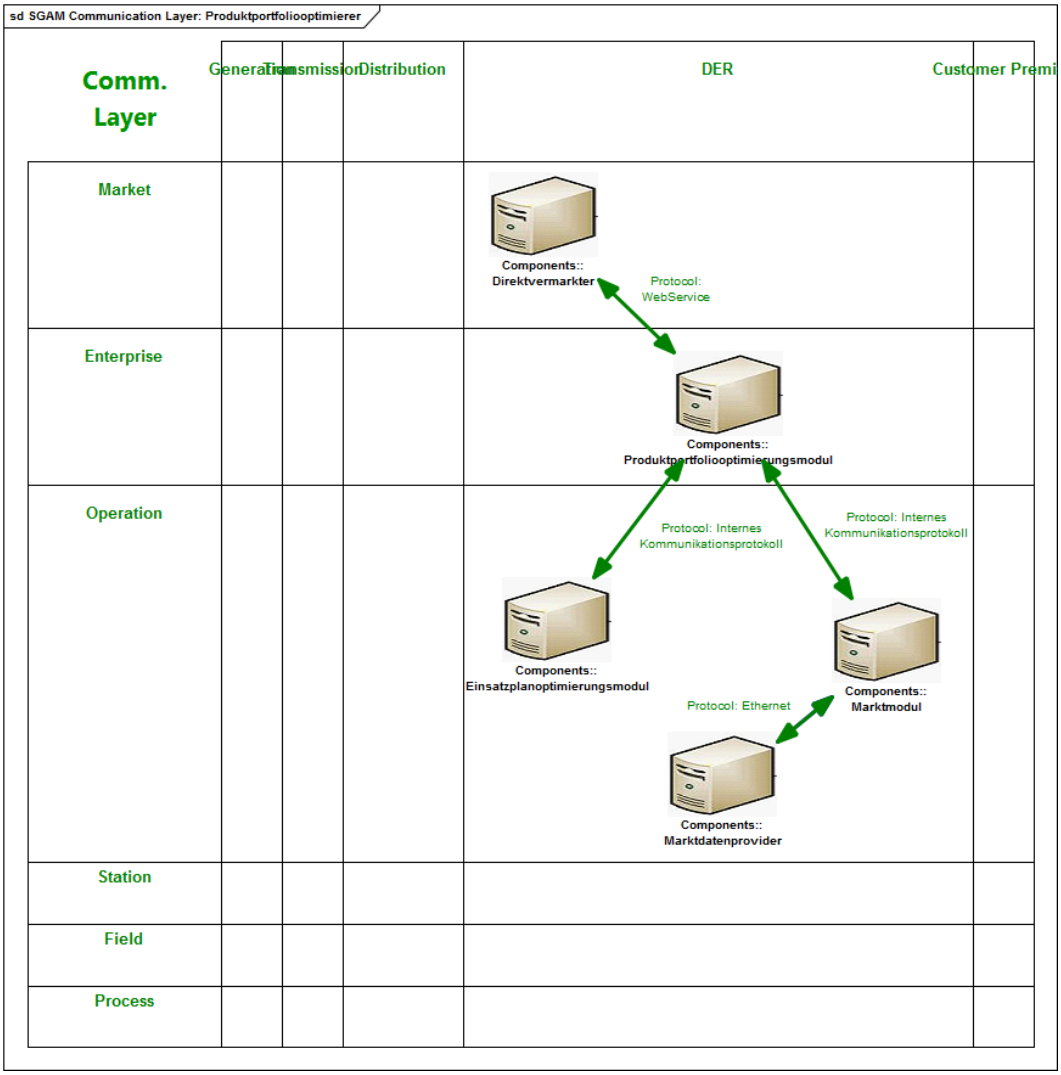


Abbildung 137: Communication Layer HLUC *Produktportfoliooptimierung*

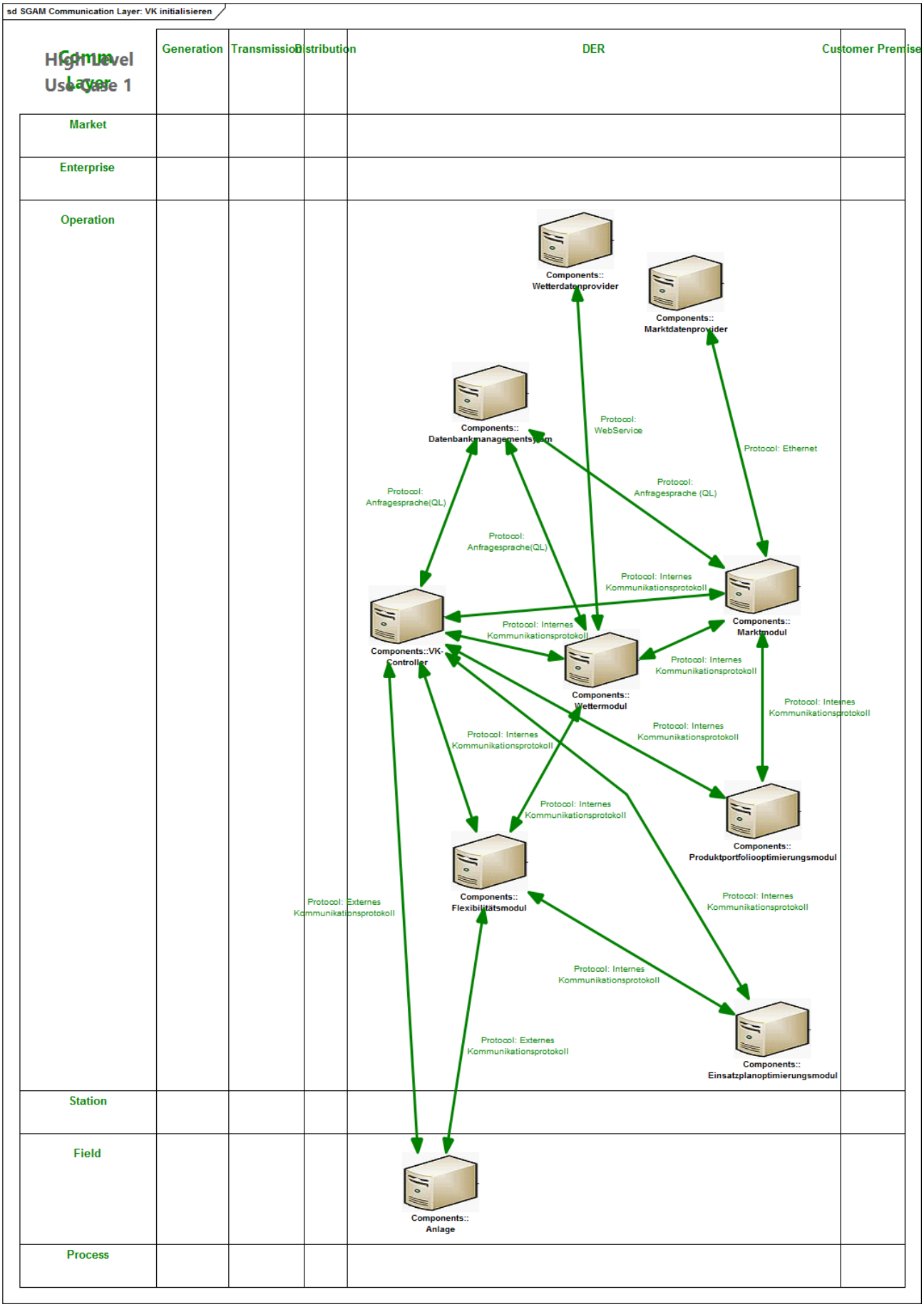


Abbildung 138: Communication Layer HLUC *VK initialisieren*

E. Benutzerhandbuch

Dieses Kapitel erklärt, wie der Benutzer *Powder* installiert und benutzt. *Powder* ist als Forschungsprojekt gedacht und arbeitet mit den historischen Markt- und Wetterdaten des Jahres 2012. Um es in der Produktion einzusetzen, sind Anpassungen nötig. Diese werden im Entwicklerhandbuch (Anhang F) beschrieben.

Powder ist plattformübergreifend in Java implementiert und wurde auf Windows, Mac und Linux getestet. Ausgeliefert werden Jar-Dateien, ein Ordner `resources` mit Konfigurationsdateien und eine Datenbankdatei. Für die Ausführung unter Windows liegen außerdem CMD-Dateien vor. Auf dem System muss Java 8 Update 40 [22] oder neuer installiert sein.

E.1. Konfigurationen

Es können verschiedene Einstellungen über Textdateien konfiguriert werden (s. Abb. 139). Dazu zählen die Stammdaten (techdata) und die Zusammensetzung des Anlagenpools (DefaultPlantConfiguration).

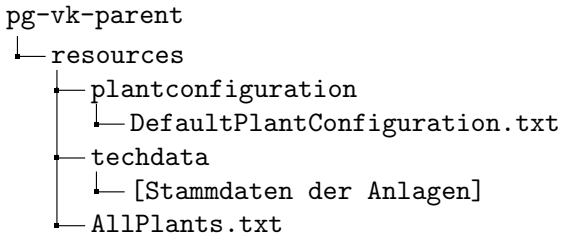


Abbildung 139: Konfigurationsdateien im `resources`-Verzeichnis

E.1.1. techdata

In dem Verzeichnis `techdata` können die Stammdaten für die Anlagen definiert werden. Dazu wird eine Textdatei mit dem folgenden Aufbau genutzt (s. Code-Ausschnitt 9):

Code-Ausschnitt 9: PV_10KW

```
1 SURFACE=100
2 EFFICIENCY=0.14
3 TILT_ANGLE=40
4 ORIENTATION_ANGLE=NW
5 GROUND=Asphalt
6 LATITUDE=52.2443
7 LONGITUDE=10.5594
8 MODUL_SURFACE=1.66
9 NOMINAL_MODUL_POWER=0.23
```

[22] http://www.java.com/de/download/

In jeder Zeile befindet sich eine Eigenschaft, deren Wert hinter einem Gleichheitszeichen angegeben wird. Eine Übersicht der Eigenschaften wird nachfolgend für BHKW (s. Tab. 13) und PV-Anlagen (s. Tab. 14) gegeben.

Eigenschaft	**Einheit**	**Beschreibung**
MIN_ELECTRICAL_POWER	KiloWatt	Minimale Leistung, wenn das BHKW läuft
MAX_ELECTRICAL_POWER	KiloWatt	Maximale Leistung
ELECTRIC_TO_THERMAL	Decimal-Percentage	Verhältnis zwischen elektrischer und thermischer Leistung
MAX_TOTAL_EFFICIENCY	Decimal-Percentage	gesamt Wirkungsgrad bezogen auf elektrische und thermische Leistung Verhältis in 0.XX Prozent
MIN_TOTAL_EFFICIENCY	Decimal-Percentage	gesamt Wirkungsgrad bezogen auf elektrische und thermische Leistung Verhältis in 0.XX Prozent
OWN_POWER_CONSUMPTION	KiloWatt	Eigenstromverbrauch für den Betrieb der Anlage
MIN_AKTIV_TIME	Minutes	minimale aktive Zeit
MIN_INACTIV_TIME	Minutes	minimale inaktive Zeit
LUBRICANT_CONSUMPTION_PER_HOUR	MilliLiter	Schmiermittelverbrauch pro Stunde
AMOUNT_OF_LUBRICANT	Liter	Fassungsvermögen in Liter (Schmiermittel)
CO_EMISSIONS_PER_HOUR	MilliGram	Erzeugtes Kohlenmonoxid als Abgase
NOX_EMISSIONS_PER_HOUR	MilliGram	Erzeugtes Stickstoffoxid als Abgase
CAPACITY_OF_BUFFER_STORAGE	Liter	Fassungsvermögen des Pufferspeichers
BUFFER_STORAGE_ENERGY_CONSUMPTION_24H	KiloWattHour	Bereitschaftsenergieverbrauch des Pufferspeichers pro Tag
MIN_TEMPERATURE_OF_BUFFER_STORAGE	DegreeCelsius	Minimaltemperatur, die das Wasser im Pufferspeicher annehmen kann.
MAX_TEMPERATURE_OF_BUFFER_STORAGE	DegreeCelsius	Maximaltemperatur, die das Wasser im Pufferspeicher annehmen kann.
MIN_PERFORMANCE_LIMIT	Decimal-Percentage	Prozent > 0, die das BHKW minimal laufen kann.

Tabelle 13: Eigenschaften von BHKW

Eigenschaft	Einheit	Beschreibung
SURFACE	SquareMeter	Gesamtfläche
EFFICIENCY	Decimal-Percentage	Wirkungsgrad
TILT_ANGLE	ArcDegree	Neigung
ORIENTATION_ANGLE	ArcDegree	Ausrichtung
GROUND	String	Name des Untergrundes[1] (Reflektionswert wird dann nachgeschaut)
LATITUDE	ArcDegree	Standortbreitengrad
LONGITUDE	ArcDegree	Standortlängengrad (0° = Süden)
MODUL_SURFACE	SquareMeter	Die Fläche eines Moduls
NOMINAL_MODUL_POWER	KiloWatt	Nennleistung eines Moduls (KWPEAK)

Tabelle 14: Eigenschaften von PV-Anlagen

E.1.2. andere Dateien

Eine Liste der auswählbaren Anlagen muss in `AllPlants.txt` angegeben werden. In dieser Datei steht pro Zeile der Name einer techdata-Datei ohne Endung (s. Code-Ausschnitt 10).

Code-Ausschnitt 10: AllPlants.txt

```
1 CHP_LPG_0KW6
2 CHP_LPG_20KW
3 CHP_LPG_50KW
4 PV_20KW
5 PV_5500KW
```

In `DefaultPlantConfiguration.txt` wird eine Standardzusammenstellung von Anlagen angegeben (Anlagenpool). Die Anlagennamen müssen mit den Dateinamen der `techdata` übereinstimmen. Dabei kann angegeben werden, wie viele Anlagen von welchem Typ dem virtuellen Kraftwerk angehören (s. Code-Ausschnitt 11). Zusätzlich kann mit `Number_of_Scedules` festgelegt werden, wie viele Flexibilitäten pro Anlage generiert werden sollen.

Code-Ausschnitt 11: DefaultPlantConfiguration.txt

```
1 Number_of_Scedules=20
2 CHP_LPG_0KW6=4
3 CHP_LPG_20KW=1
4 CHP_LPG_50KW=2
5 PV_20KW=3
6 PV_5500KW=1
```

[1] Für `GROUND` gibt es die Typen: Gras, Rasen, UnbestellteFelder, BlosserBoden, Schotter, BetonSauber, BetonVerwittert, ZementSauber, Asphalt, Waelder, Wasserflaeche und Schneedecke.

E.2. Ausführung

Powder kann in zwei Versionen ausgeführt werden. Zum einen existiert eine Core-Version, die nur in der Konsole ausgeführt wird. Die Eingabe von Werten und die Ausgabe der Ergebnisse finden somit auch in der Konsole statt. Die zweite Version ist die GUI-Version. Hier können sämtliche Interaktionen mit dem Benutzer über die graphische Oberfläche stattfinden.

Zu beachten ist, dass beim ersten *Powder*-Start und somit beim ersten Anmeldeversuch der Anlagen über OPC UA, Zertifikate erstellt und nicht akzeptiert werden. Daher können beim ersten *Powder*-Optimierungsdurchlauf keine Fahrpläne an die Anlagen versendet werden. Die Zertifikate liegen im Pfad `PKI/CA/rejected` und müssen in `PKI/CA/certs` verschoben werden. Sollten sich keine Zertifikate im `rejected`-Ordner befinden, müssen die Lese- und Schreibrechte auf Betriebssystemebene angepasst werden. Falls beim ersten Ausführen keine Zertifikate entstanden sind, muss *Powder* nochmals ausgeführt werden. Es müssen zwei Zertifikate im `certs`-Ordner vorhanden sein. Falls nur ein Zertifikat erstellt wurde, muss die *Powder*-Optimierung nochmals gestartet werden und das entstandene Zertifikat ebenfalls zu `certs` verschoben werden. Danach kann die Optimierung mit *Powder* normal durchgeführt werden.

E.2.1. Ausführung der Core-Version

Die Core-Version kann gestartet werden, indem die `powder-core.cmd` ausgeführt wird. Anschließend öffnet sich ein Fenster und es werden insgesamt drei Aufforderungen gestellt. Um zur nächsten Aufforderung zu gelangen, muss die Eingabe durch Drücken der Eingabetaste bestätigt werden.

In der ersten Aufforderung wird nach einem Datum gefragt, für das die Optimierung durchgeführt werden soll. Das Datum muss das Format `yyyyMMdd` besitzen. Eine mögliche gültige Eingabe wäre demnach `20120506`, um den 6. Mai 2012 auszuwählen. Es kann zudem ein voreingestelltes Datum ausgewählt werden, indem nichts eingegeben und die leere Eingabe durch Drücken von Enter bestätigt wird.

Die zweite Aufforderung fragt nach einem Seed. Dieser ist nur zur Fehlerbehebung relevant und kann daher, durch drücken der Eingabetaste, übersprungen werden.

Die dritte Aufforderung betrifft die Auswahl der Anlagen, die zur Optimierung benutzt werden sollen. Hier muss der Name einer `txt`-Datei eingegeben werden, die sich im Ordner `resources/plantconfiguration` befindet. Sollte keine Eingabe erfolgen, wird automatisch die Datei `Default-PlantConfiguration.txt` ausgewählt.

E.2.2. Ausführung der GUI-Version

Die GUI-Version kann gestartet werden, in dem die `powder-gui.cmd` gestartet wird. Auf der Start-Seite (Abbildung 142) werden beim ersten Start zunächst zufällige Diagramme angezeigt. Sobald Optimierungsergebnisse vorhanden sind, wird hier das letzte Ergebnis angezeigt.

Zur Vorbereitung der ersten Optimierung ist es ratsam, die teilnehmenden Anlagen auszuwählen. Dieser Schritt kann übersprungen werden, wenn die Anlagenkonfiguration der letzten Optimierung beibehalten werden, oder eine der gespeicherten Anlagenkonfigurationen ausgewählt werden soll.

Jene Anlagenkonfiguration der letzten Optimierung wird beim Starten der GUI automatisch als zu verwendende Anlagenkonfiguration gesetzt.

Über die VK-Settings (Abbildung 140) können die auszuwählenden Anlagen zu erst durch einen Klick markiert und dann über die Pfeile zur Auswahl hinzugefügt werden.

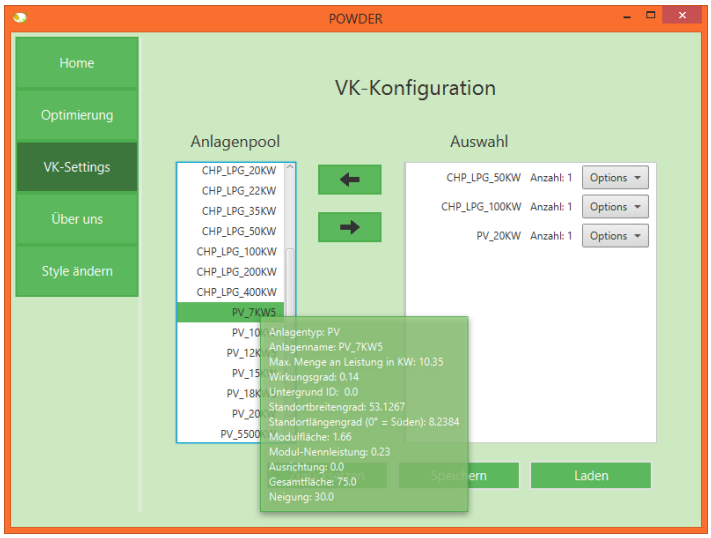


Abbildung 140: VK-Settings

Die Stammdaten zur Anlage können über ein verweilen der Maus auf dem Namen als Tooltip angezeigt werden. Wenn mehrere Anlagen eines BHKW- oder PV-Typs vorhanden sein sollen, müssen sie entweder durch mehrfaches Anklicken des Pfeils oder über das Options-Menü der Anlage im Auswahlfeld hinzugefügt werden. Auf dieselbe Weise können auch Anlagen entfernt werden. Es muss darauf geachtet werden, dass die teilnehmenden Anlagen zu einer Gesamtleistung von mindestens 100kW führen. Die Anlagenkonfiguration kann außerdem gespeichert oder eine bereits vorhandene geladen werden. Sollen alle Anlagen aus der Auswahl entfernt werden, kann dies über den „Zurücksetzen“-Button geschehen.

Die Optimierung kann auf der Optimierungsseite gestartet werden.

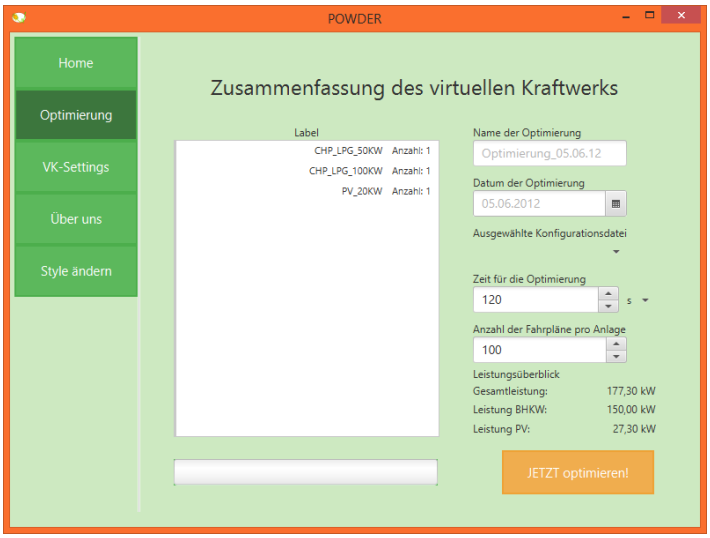


Abbildung 141: Optimierung

Die teilnehmenden Anlagen sind in dem linken Feld visualisiert. Es muss ein Name für die Optimierung bestimmt werden und ein Datum ausgewählt werden, für die eine Optimierung stattfinden soll. Es kann die Zeit, die die Optimierung dauern darf und die Anzahl der Fahrpläne pro Anlage eingestellt werden. Hier können außerdem gespeicherte Anlagenkonfigurationen ausgewählt werden. Wenn die Gesamtleistung größer als 100kW ist, kann die Optimierung gestartet werden. Danach wird der Optimierungsprozess über einen Fortschrittsanzeige in Form eines Balkens angezeigt. Die Benutzung der GUI ist während der Optimierung gesperrt. Sollte der Optimierungsvorgang abgebrochen werden, muss die zuletzt erstellte Datei im Ordner `resources/results/.info` gelöscht werden.

Die Ergebnisse der Optimierung werden auf der Startseite (Abbildung 142) visualisiert.

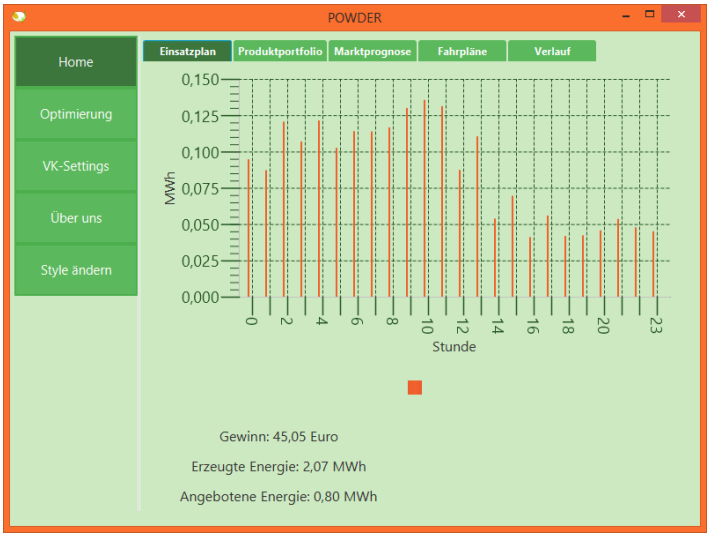


Abbildung 142: Startansicht

Im ersten Reiter „Einsatzplan“ wird der Einsatzplan in stündlicher Auflösung als Graph dargestellt. Unter dem Graphen kann der Gewinn, die erzeugte Energie und die angebotene Energie entnommen werden. Im Reiter „Produktportfolio“ kann das am Markt angebotene Produktportfolio angesehen

werden. Im unteren Bereich des Fensters kann unter „Diagrammauswahl“ zwischen „Flächendiagramm“ und „gestapeltes Balkendiagrammäusgewählt werden. Im Reiter „Marktprognose“ wird die Marktprognose angezeigt, die zur Erstellung des Produktportfolios benutzt wird. Hier werden sowohl die Blockpreise, als auch die Stundenpreise visualisiert. Im Reiter „Fahrpläne“ werden zunächst die Fahrpläne aller Anlagen angezeigt. Mittels des Buttons „Anlage auswählen“ im unteren, rechten Bereich des Fensters, kann auch der Fahrplan zu einer Anlage angezeigt werden oder zur Übersicht der Fahrpläne aller Anlagen zurückgekehrt werden. Der Graph ist in 15-Minuten-Schritten für einen Tag unterteilt, sodass 96 Balken angezeigt werden. Sollte der Graph nicht korrekt dargestellt werden, muss das *Powder*-Fenster verbreitert werden. Im unteren, linken Bereich des Fensters wird außerdem die maximale und die erzeugte Leistung dargestellt. Im Reiter „Verlauf“ können verschiedene erfolgreiche Optimierungen ausgewählt und in den anderen Reitern angeschaut werden.

E.3. Ausgabe

Powder erstellt als Ergebnis eines Durchlaufs ein Java-Objekt `OptimizationResult`. Es ist wie folgt aufgebaut:

- Die Methode `getProfit()` berechnet den Gesamtgewinn, indem sie von den Markterlösen und Boni die Kosten subtrahiert.
- Das Objekt `ProductPortfolio` enthält eine `Collection` von `MarketProducts`. Ein `MarketProduct` enthält folgende Werte:
  - `marketProductType`: Das EPEX-Spot Produkt
  - `energyAmount`: Menge an zu verkaufenden Megawattstunden (MWh)
  - `predictedPrice`: Der vom Marktmodul vorhergesagte Preis
  - `offerPrice`: Der von *Powder* vorgeschlagene Verkaufspreis
- Das Objekt `OperationSchedule` enthält eine `HashMap` aus `PlantIDs` und `Schedules`. Ein `Schedule` gehört also zu einem Kraftwerk und enthält folgende Werte:
  - `plantPowerPerStep`: Fahrplan in Form von einer KiloWatt-Angabe für jede Viertelstunde des kommenden Tages
  - `costsPerStep` Kosten in Euro für jede Viertelstunde
  - `rewardsPerStep` Vergütung in Euro (z. B. durch EEG-Umlage) für jede Viertelstunde
  - `overallCosts` Gesamtkosten in Euro
  - `overallRewards` Gesamtvergütung in Euro

Die GUI-Version von *Powder* zeigt diese Objekte in Form übersichtlicher Diagramme an.

F. Entwicklerhandbuch

Dieses Kapitel erklärt, wie *Powder* weiterentwickelt werden kann. Für ein grundlegendes Verständnis der Funktionsweise wird auf Abschnitt 6 verwiesen.

Das Java-Projekt *Powder* wurde mit dem Build-Management-Tool Maven strukturiert. Um es in der Entwicklungsumgebung IntelliJ IDEA zu öffnen, muss im Menü *File* → *New* → *Project from Existing Sources* gewählt werden. Dann muss *Import project from external model: Maven* gewählt werden. Schlussendlich muss unter *File* → *Settings* → *Plug-ins* → *Browse repositories* die Erweiterung Lombok installiert werden.

Um es in der Entwicklungsumgebung Eclipse zu öffnen, muss im Menü *File* → *Import* → *Maven* → *Existing Maven Projects* gewählt werden. Nun muss der Ordner `pg-vk-parent` gewählt werden. Dieser Vorgang muss für die direkten Unterordner von `pg-vk-parent` wie `databaseconnection`, `shared` etc. wiederholt werden.

F.1. Projekt auf Server ausführen

Nachfolgend wird kurz beschrieben, welche Schritte getan werden müssen, damit *Powder* auf einem Linux-Server läuft. Eine ausführlichere Beschreibung ist in Unterabschnitt 6.3 zu finden.

Auf dem Server muss Java und der X-Server `xvfb` installiert sein. Dieser muss mit dem Befehl `nohup Xvfb :7 -ac -screen 0 1024x768x24 &` gestartet werden. Dann muss im Ordner `pg-vk-parent` der Befehl `mvn install` und zudem in allen Modul-Ordnern der Befehl `mvn compile` ausgeführt werden. Dies erzeugt für jedes Modul eine Jar-Datei. Sie muss auf den Server kopiert werden. Dort muss dann `export DISPLAY=:7` und `nohup java -jar powder.jar &` ausgeführt werden. Die Ausgabe des Projekts wird in der Datei `nohup.out` geschrieben.

Für die interne Kommunikation per RMI müssen die `ModulLocation` in der Klasse `UserSpecificModulNetworkSettings` auf `Server` gesetzt werden (s. Abschnitt 6.3).

Eine einfachere Variante ist, das GIT auf dem Server zu pullen und dort die notwendigen Maven-Befehle auszuführen. Dies wird durch das Shell-Skript `pullAndBuild.sh` unterstützt. Nach dem Pullen werden mit dem Skript `createJarWithDependencieForAllModules.sh` für alle Module jar-Dateien erzeugt und in die entsprechenden Verzeichnisse verschoben, damit der korrekte Zugriff auf die Ressourcen gewährleistet werden kann (s. Code-Ausschnitt 12).

Code-Ausschnitt 12: createJarWithDependencieForAllModules.sh

```
1  compileAndCopy() {
2    cd $1
3    mvn clean compile assembly:single
4    mv ./target/*$JAR_NAME_ENDING ../../export-jar/
5    cd ..
6  }
7
8  export JAR_NAME_ENDING=-jar-with-dependencies.jar
9
10 rm -r resources #resources ordner im export-jar ordner wird gelöscht
```

```
cd ../pg-vk-parent
rm -r resources #resources ordner im projektverzeichnis wird gelöscht

mvn install -Dmaven.test.skip=true

compileAndCopy registry
compileAndCopy market
compileAndCopy flexibility
compileAndCopy optimization
compileAndCopy weather
compileAndCopy evaluation
compileAndCopy vpp

cp -r resources ../export-jar
```

Mit dem Skript `startAll.sh` (s. Code-Ausschnitt 13) werden alle Module in einzelnen Prozessen gestartet. Mit `stop.sh` (s. Code-Ausschnitt 14) werden diese Prozesse wieder beendet.

Code-Ausschnitt 13: startAll.sh

```
nohup Xvfb :7 -ac -screen 0 1024x768x24 &
export DISPLAY=:7

java -jar registry-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
sleep 1
java -jar vpp-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
java -jar flexibility-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
java -jar optimization-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
java -jar market-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
java -jar weather-0.0.1-SNAPSHOT-jar-with-dependencies.jar
```

Code-Ausschnitt 14: stop.sh

```
rm Server.tmp
```

Eine umfangreichere Beschreibung zur Ausführung auf dem Server wird im Kapitel 6.3 gegeben.

F.2. Aktualisierung der Markt- und Wetterdaten

Die historischen Marktdaten und Gaspreise können mit Hilfe mitgelieferter Scraper von der EPEX-Spot Website abgerufen werden. Die Scraper sind jeweils eine einzelne Python-Datei im Ordner `crawler`. Am Anfang der Dateien kann der abzurufende Zeitraum eingestellt werden. Ein Durchlauf mittels `python3 crawl_electricity_prices.py` in einer Kommandozeile erzeugt die Dateien `blocks.csv` und `stunden.csv`. Diese Dateien können durch Ausführen des Java-Projektes `datenparser` in die Datenbank von *Powder* importiert werden. Um neue Wetterdaten zu importieren, empfiehlt die Projektgruppe, den Datenparser an das Format der zu importierenden CSV-Datei anzupassen.

F.3. Konfigurationen

In diesem Abschnitt werden Einstellungen beschrieben, die im Code gesetzt werden können. Dieses Unterkapitel ist eine Ergänzung zu dem Konfigurationskapitel im Benutzerhandbuch (s. Abschnitt E.1)

F.3.1. Auswahl der Optimierer

In der `VPPMain` kann der Methode `startOptimization` eine `PowderRunConfiguration` übergeben werden, die beschreibt, auf welche Weise die Optimierung durchgeführt werden soll (s. Tab. 15). In der `OptimizerDescription` wird der Name des Optimierers und die Menge der für diesen notwendigen Parametern gespeichert.

Variable	**Beschreibung**
`decoderType`	Gibt an welcher Decoder genutzt werden soll (`JB_DECODER` oder `EUCLID_DECODER`)
`timeForOptimization`	Die Zeit, die für die Optimierung maximal genutzt werden darf
`numberOfSchedulesPerFlexibilty`	Anzahl der Fahrpläne, die pro Anlage generiert werden
`innerOptimizerDescription`	Beschreibung des inneren Optimierers
`outerOptimizerDescription`	Beschreibung des äußeren Optimierers

Tabelle 15: Parameter der `PowderRunConfiguration`

F.3.2. LoggerVKSettings

Für Textausgaben wird der Logger LVK genutzt. In der Klasse `LoggerVKSettings` wird eingestellt, welche Ausgaben angezeigt werden sollen. Dazu kann den in `LVK` definierten Variablen für die Ausgaben jeweils ein `log4j.Level` zugewiesen werden. Standardmäßig wird das Level `ERROR` gesetzt, damit nur wichtige Ausgaben erscheinen. Eine ausführlichere Beschreibung des Loggers wird in Kapitel 5.9 gegeben.

Außerdem können einige Debug-Einstellungen gesetzt werden. Mit `NO_INPUTS` kann eingestellt werden, ob beim Ausführen des Projekts (ohne GUI) Eingaben für Startdatum usw. gemacht werden müssen oder jeweils Standardwerte genutzt werden (s. Abschnitt E.2.1). Mit `START_OPC_SERVER_AND_CLIENT` kann eingestellt werden, ob die OPC UA-Kommunikation genutzt werden soll (s. Abschnitt 6.2.3).

F.3.3. OPC

Der Aufbau der CIM für die Anlagen wird im Kapitel Externe Kommunikation: Anlagen beschrieben.

Da das Projekt auf der Prosys OPC UA Evaluationsversion aufbaut, sei an dieser Stelle auch auf deren Dokumentation verwiesen.

Mit dem UaModeler (s. Abschnitt 5.12) kann aus den Common Information Modellen der Anlagen eine `NodeSet.xml` erzeugt werden. Dazu müssen im UaModeler die CIM nachmodelliert werden, da ein Import nicht möglich ist.

Die `NodeSet.xml` muss in der `codegen.properties` des Code Generators von Prosys angegeben werden, damit sie genutzt werden kann. Außerdem kann dort der `package`-Name sowie der Exportpfad für Interfaces und Implementierungen definiert werden. Danach wird die `build.xml` mit dem Werkzeug Ant ausgeführt, um den Code zu generieren.

Im OPC UA-Server kann über `securityMode` eingestellt werden, welche Verschlüsselungsverfahren akzeptiert werden sollen. Im `PlantClient` kann ein konkretes Verschlüsselungsverfahren angegeben werden. Zudem können Benutzername und Passwort festgelegt werden, falls diese für das gewählte Protokoll notwendig sind.

F.4. Evaluation

In diesem Abschnitt wird beschrieben, wie die Evaluation funktioniert, bzw. welche Einstellungen für die Durchführung der Evalaution möglich sind. Für eine genaue Darstellung der Funktionsweise sei an dieser Stelle auf den Abschnitt 6.10 verwiesen.

Ablauf der Evaluation Die Evaluation führt *Powder* für spezifisch gewählte Szenarien aus. Diese Szenarien sind durch sog. ParameterSets definiert, welche eine Zusammenstellung aus Flexibilitäten, Marktprognose, Einsatzplan (falls nur das Produktportfolio optimiert werden soll) und der genauen Anzahl an Anlagen aller parametrisierter Anlagentypen (von PV- und BHKW-Anlagen) beinhaltet.

Nach Ausführung der Evaluation finden sich die Ergebnisse in der Datei `OverallEvaluationResults.csv`, welche sich in den Ressourcen unter `evaluation/results/` befindet.

Die Ergebnisse bestehen aus der durchschnittlichen absoluten Fitness, der Lösungsqualität (im Vergleich zur optimalen Lösung), der Laufzeit, der Anzahl der Fitnessfunktionsaufrufe insgesamt und bis das beste Ergebnis gefunden wurde. Ebenfalls enthalten ist die Standardabweichung der Fitness über die Anzahl der Ausführungen und ob das Ergebnis auf den vorgegebenen Anlagen realisierbar ist.

Einstellungsmöglichkeiten vor dem Starten der Evaluation Vor dem Starten können in den Klassen `EvaluationMain` und `EvaluationParameterSetCreator` verschiedene Einstellungen vorgenommen werden.

Die EvaluationMain bietet Einstellungen für den allgemeinen Durchlauf der Evaluation:

- DECODER_TYPE legt den Dekodierer fest, der für die Durchführung genutzt werden soll. Dabei gibt es zwei Möglichkeiten:
  - DecoderTypes.JB_DECODER nutzt den Supportvektordekodierer, welcher aus den vorliegenden Flexibilitäten neue berechnen kann. Diese Flexibilitäten sind nach ihrer Berechnung durch die Anlagen realisierbar
  - DecoderTypes.EUCLID_DECODER bildet Fahrpläne auf den euklidisch nächsten Fahrplan innerhalb der gegebenen Fahrpläne ab
- MAX_COMPUTATION_TIME_PER_OPTIMIZER setzt die Sekunden fest, die jeder Optimierer pro Lauf erhält (wird immer komplett ausgenutzt)

- NUMBER_REPETITIONS gibt an, wie häufig jeder Optimierer mit jedem ParameterSet ausgeführt wird (zur Berechnung der Standardabweichung der Optimierer)
- DESIGN_TYPE gibt an, welchen Designtyp der EvaluationParameterSetCreator für die Erstellung der ParameterSets verwenden soll. Dabei gibt es zwei Möglichkeiten:
  - DesignType.LATIN_HYPERCUBE nutzt das LatinHyperCube Verfahren, welches auch für wenige Szenarien, die Parameter möglichst gut über den Raum verteilt
  - DesignType.FULL_FACTORIAL erzeugt für den Parameterraum Szenarien, die diesen gerastert komplett abdecken (die Rasterfeinheit ist gleich NUMBER_INTERVALS)
- START_DATE Datum an welchem die Optimierungen stattfinden
- END_DATE falls die fünfte Option beim Ausführen der `EvaluationMain` genutzt wird, ist dies das Enddatum des Zeitraums und START_DATE der Beginn
- DESIGN_TYPE ermöglicht die Auswahl zwischen Latin Hypercube und vollfaktoriellem Design für die Generierung der durchzuführenden Szenarien

Die Klasse EvaluationParameterSetCreator erstellt mittels des in der EvaluationMain angegebenen Designtyps ParameterSets (Szenarien) für den mit den folgenden Parametern aufgespannten Raum:

- NUMBER_INTERVALLS Die Feinheit des Designtyps. Für jedes Intervall wird (beim vollfaktoriellen Design für jede der vier Dimensionen, ansonsten nur) ein weiteres ParameterSet erstellt
- MIN_PLANT_COUNT / MAX_PLANT_COUNT Grenzen für die Anzahl der Anlagen
- MIN_POWER_RATIO_PLANTTYPE / MAX_POWER_RATIO_PLANTTYPE Obere und untere Grenzen für den Anteil der erzeugten Energie durch BHKWs (s. Abschnitt 6.10.3)
- MIN_SCHEDULE_COUNTS_CHP / MAX_SCHEDULE_COUNTS_CHP Fahrpläne, die je BHKW erzeugt werden (Flexibilitätsraum bei Nutzung des euklidischen Dekodierers bzw. Trainingsmenge des Supportvektordekodierers)
- MIN_OVERALL_POWER / MAX_OVERALL_POWER Leistung des Virtuellen Kraftwerks

Einstellungsmöglichkeiten beim Starten der Evaluation Die Evaluation kann über Ausführung der Klasse `EvaluationMain.java` gestartet werden. Auf dem Server ist dies nach der Bereitstellung durch Ausführen des Shell Scripts `EvaluationStart.sh` ebenfalls möglich.

Folgende Ausführungsvarianten sind entwickelt worden:

1. Erstellung der ParameterSets: Hier werden ParameterSets generiert, je nach Einstellungen in der Klasse `EvaluationParameterSetCreator`
2. Kombinierten Optimierer ausführen: Der `CombinedPPOEPOLPOptimizer` berechnet für alle vorliegenden ParameterSets das optimale Ergebnis und speichert es anschließend ab

3. Innere Optimierer ausführen: Mit dem im Parameterset enthaltenen Einsatzplan wird ein Produktportfolio von allen Produktportfoliooptimierern (`RecursiveIntervallOptimizer` und `LinearProgrammingPPOSolver`) berechnet und diese Durchläufe anschließend evaluiert

4. Äußere Optimierer ausführen: Für die ParameterSets werden die Einsatzplanoptimierer (in Kombination mit dem besten Produktportfoliooptimierer, also dem `RecursiveIntervall-Optimizer`) ausgeführt und evaluiert

5. Optimierer für mehrere Tage ausführen: Für alle 30 Tage, die seit dem START_DATE vergangen sind, wird einmal der kombinierte Optimierer ausgeführt und dessen Ergebnisse gespeichert. Am Ende wird der Mittelwert der Fitness geloggt

G. Seminarband

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik – Abteilung Umweltinformatik

Seminarband der Projektgruppe Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks

Sommersemester 2015

Vorwort

Dieser Seminarband fasst die im Rahmen der Seminarphase der Projektgruppe *Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks* erstellten Seminarbeiträge zusammen.

Oldenburg,
September 2015

Inhaltsverzeichnis

Referenten

Prof. Dr. Michael Sonnenschein (Editor)
sonnenschein@informatik.uni-oldenburg.de

Dr. Ute Vogel (Editor)
vogel@informatik.uni-oldenburg.de

Dr. Christian Hinrichs (Editor)
hinrichs@informatik.uni-oldenburg.de

Stephan Balduin
stephan.balduin@uni-oldenburg.de

Dierk Brauer
dierk.brauer@uni-oldenburg.de

Lars Elend
lars.elend@uni-oldenburg.de

Stefanie Holly
stefanie.holly@uni-oldenburg.de

Jan Korte
jan.korte@uni-oldenburg.de

Carsten Krüger
carsten.krueger@uni-oldenburg.de

Almuth Meier
almuth.meier@uni-oldenburg.de

Frauke Oest
frauke.oest@uni-oldenburg.de

Immo Sanders-Sjuts
immo.sanders-sjuts@uni-oldenburg.de

Torben Sauer
torben.sauer@uni-oldenburg.de

Marco Schnieders
marco.schnieders@uni-oldenburg.de

Robert Zilke
robert.zilke@uni-oldenburg.de

Kapitel 1
Virtuelle Kraftwerke

Dierk Brauer

Zusammenfassung In dieser Ausarbeitung soll zunächst darauf eingegangen werden, wozu virtuelle Kraftwerke (VK) benötigt werden und was ein VK ist. Dann wird eine Abgrenzung zu Microgrids vorgenommen. Als nächstes wird auf die Steueraufgaben eines VK eingegangen und ein zentraler sowie dezentraler Ansatz zur Steuerung vorgestellt. Darauf folgend wird das bereits abgeschlossene Projekt Kombikraftwerk 2 vorgestellt, in dem gezeigt wurde wie Deutschland Mitte des 21. Jahrhunderts ausschließlich durch erneuerbare Energien mit elektrischer Energie versorgt werden könnte. Mit Next Pool wird außerdem ein kommerzielles VK, welches es Kleinerzeugern ermöglicht Energie an Großabnehmer zu verkaufen, vorgestellt. Als letztes wird eine Zusammenfassung und Fazit gegeben.

1.1 Einleitung und Motivation

In diesem Kapitel wird zunächst eine Motivation für die Verwendung von virtuellen Kraftwerken (VKs) , sowie ein kurzer Überblick über die Funktion einer Strombörse gegeben. Danach wird ein kurzer Überblick über den Aufbau dieser Ausarbeitung gegeben.

In der Energieerzeugungs-Branche ist seit geraumer Zeit ein Wandel zu beobachten. Zurzeit ist es zwar noch so, dass der Großteil der elektrischen Energie in Deutschland durch große Kraftwerke, welche Energie durch fossile Energieträger gewinnen, versorgt wird. Jedoch gewinnen erneuerbare Energien (EE) in den letzten Jahren immer mehr an Bedeutung [6]. Auch ist es erklärtes Ziel der deutschen Bundesregierung, die Energieproduktion durch EE voranzutreiben, so dass in Zukunft der Großteil der Energie in Deutschland durch EE produziert werden soll [4].

Carl von Ossietzky Universität Oldenburg
E-mail: dierk.brauer@uni-oldenburg.de

Ein Problem das dadurch entsteht ist es, dass durch Windenergie und Photovoltaik (PV) durch ungeplante Schwankungen von Sonnen- und Windkraft auch Schwankungen im Netz entstehen. Durch Außerbetriebnahme von konventionellen Kraftwerken werden zudem Möglichkeiten zum Ausgleichen der Schwankungen reduziert.

Es gibt immer mehr kleine Anlagen die elektrische Energie erzeugen, welche häufig auch von Privatbesitzern betrieben werden. Um diese Energien möglichst gewinnbringend zu verkaufen, ist ein Handel an der Strombörse erstrebenswert. Bei der Strombörse handelt es sich um einen organisierten Markt für elektrische Energie, in der Energie-Blöcke ähnlich wie Wertpapiere an der Börse gehandelt werden. Diese Energie-Blöcke verpflichten den Verkäufer dazu über einen gewissen Zeitraum eine gewisse elektrische Leistung zur Verfügung zu stellen (z.B. muss eine kontinuierliche Leistung von einem MW von 0 bis 24 Uhr in das Stromnetz eingespeist werden). Die Leistung die ein kleines Kraftwerk zur Verfügung stellen kann, ist jedoch in aller Regel viel zu gering, um damit an der Strombörse handeln zu können. Des Weiteren sind z.B. PV und Windenergie von Variablen, die nicht zu beeinflussen sind (Wetter) abhängig, wodurch es schwer wird eine Leistung kontinuierlich zur Verfügung zu stellen.

Folgend nun der Aufbau dieser Arbeit. In Kap. 1.2 wird zunächst darauf eingegangen, was virtuelle Kraftwerke (VKs) sind und wie mithilfe von diesen eine Vermarktung der Energie von Kleinerzeugern an der Strombörse ermöglicht werden kann. In Kap. 1.3 werden außerdem Microgrids vorgestellt und diese gegenüber VKs abgegrenzt. Daraufhin wird in Kap. 1.4 auf die Steuerungsaufgaben in einem VK eingegangen. Es werden außerdem ein zentraler sowie ein dezentraler Ansatz zur Steuerung eines VK vorgestellt. Danach wird in Kap. 1.5 auf das Forschungsprojekt Kombikraftwerk 2 eingegangen, welches sich mit dem Thema VK und wie damit benötigte Netzdienstleistungen zu Verfügung gestellt werden können, beschäftigt hat. Als nächstes wird in Kap. 1.6 auf ein kommerzielles VK eingegangen, welches es Kleinerzeugern ermöglicht sich diesem anzuschließen und die produzierte Energie auf verschiedene Art und Weise zu verkaufen. Als letztes wird in Kap. 1.7 eine Zusammenfassung und ein Fazit über die Inhalte dieser Ausarbeitung gegeben.

1.2 Was ist ein virtuelles Kraftwerk?

Ein virtuelles Kraftwerk (VK) ist ein dezentraler Verbund von Stromerzeugern welche über ein Energie Management System (EMS) gesteuert und überwacht werden können [9]. In einem EMS können dabei eine Vielzahl von Anlagen verwaltet werden. Ein VK muss nicht auf bestimmte Arten von Stromerzeugern beschränkt sein, denn es können prinzipiell jede Art von Stromerzeugern an ein VK angeschlossen werden.

Der Begriff ”virtuell” soll nicht etwa andeuten, dass keine reale elektrische Energie erzeugt wird. Vielmehr ist dieser darauf bezogen, dass es sich nicht um ein großes,

sondern um viele kleine Kraftwerke handelt. Im Zusammenschluss können trotzdem große Leistungen erzeugt werden, wie auch in Kap. 1.6 beispielhaft gezeigt wird.

Ein VK ermöglicht verschiedenen Kleinerzeugern sich zusammenzuschließen und so eine größere, kontinuierliche Energieerzeugung. Schwankungen können durch verschiedene Anlagentypen besser ausgeglichen werden. So kann jeder Besitzer einer Anlage auch geringe Leistungen an der Strombörse verkaufen. Die große Anzahl der Anlagen machen den Einsatz flexibler und ermöglichen eine bessere Ausfallkompensation.

1.3 Microgrids

In diesem Kapitel werden Microgrids vorgestellt, welche ein ähnliches Konzept wie VKs haben. Da diese beiden Begriffe jedoch nicht zu verwechseln sind, wird zur Abgrenzung vor allem auf die Unterschiede der beiden Konzepte eingegangen. In dem gesamten Kapitel wird immer wieder Bezug auf Nikos Hatziargyriou [5] genommen.
Ein Microgrid ist, ähnlich wie ein VK, ein Verbund von Energiequellen, jedoch zusätzlich auch von Energieverbrauchern. Anders als VKs, welche immer ein Teil des Hauptstromnetzes sind, sind Microgrids als einzelne und selbstständige Stromnetze anzusehen. In der Regel sind sie nichtsdestotrotz mit dem Hauptstromnetz verbunden. Ausnahmen sind beispielsweise auf Inseln zu finden. Microgrids haben, wenn sie an das Hauptstromnetz angeschlossen sind, die Möglichkeit des ”Islanding”. Das bedeutet sie können sich vollständig von dem Hauptstromnetz isolieren. Diese Funktion kann beispielsweise bei Stromausfällen sehr nützlich sein, um innerhalb des Microgrids trotzdem ein stabiles Netz gewährleisten zu können.

Im Gegensatz zu VKs, bei denen die einzelnen Stromerzeuger über weite Gebiete verteilt sein können, sind Microgrids auf ein eher kleines geographisches Gebiet begrenzt. Dieses ist auch bedingt dadurch, dass Microgrids eine geringe Netzauslastung anstreben. Eines der Hauptkonzepte ist es daher, die Distanz zwischen Energieerzeugung und Energieverbrauch möglichst gering zu halten.

Bei Microgrids wird typischerweise immer eine Möglichkeit um Energie zu speichern benötigt. Andernfalls ist es häufig nicht Möglich unabhängig von dem Hauptstromnetz zu jedem Zeitpunkt eine ausreichende Energieversorgung zu gewährleisten. Gerade im Zusammenhang mit EE, welche häufig von nicht zu beeinflussenden Variablen (wie z.B. dem Wetter) abhängen, werden Microgrids ohne Energiespeicher-Möglichkeiten vor kaum überwindbare Probleme gestellt.

In einem Microgrid werden zwischen wenigen KW und mehreren MW elektrische Energie erzeugt bzw. Verbraucht. Dahingegen können VKs auch durchaus Leistungen im GW-Bereich erzeugen (siehe dazu auch Kap. 1.6) und somit durchaus mit großen Atomkraftwerken mithalten.

1.4 Steuerungsaufgaben im VK

In diesem Kapitel wird beschrieben, wie die Steuerungsaufgaben in einem VK realisiert werden können. Dabei wird in Kap. 1.4.1 zunächst auf die Möglichkeit der zentralen Steuerung durch ein EMS eingegangen. In Kap. 1.4.2 hingegen wird ein dezentraler Ansatz vorgestellt, in dem einzelne Komponenten jeweils durch einen Agenten gesteuert werden.

1.4.1 Zentrale Steuerung

Im Folgenden wird das Konzept eines VK mit zentraler Steuerung durch ein EMS, wie von Lombardi et al. [9] vorgestellt, beschrieben. Das EMS ist mit allen Komponenten des VK verbunden und kann Daten, wie z.B. aktuelle Energie-Produktion, von diesen abrufen. Weiterhin kann das EMS die Komponenten aber auch Steuern, um z.B regelbare Kraftwerke hoch-, oder herunterzufahren. Außerdem kann das EMS den Energiefluss kontrollieren und z.B. überschüssige Energie zu Speicherkraftwerken zu lenken. Auch Flaschenhälse im Stromnetz können so ggf. umgangen werden. Das EMS verfolgt dabei Ziele, wie z.B. Minimierung von Produktions-Kosten, Minimierung von Treibhausgasen oder Maximierung des Profits. Entsprechend können Fahrpläne erstellt werden, welche wiederum Pläne für jedes regelbare Kraftwerk beinhalten.

Gerade wenn ein Großteil der Energie durch wetterabhängige Energieerzeuger (PV, Windenergie) bereitgestellt wird, kann es leicht zu starken unvorhersehbaren Schwankungen in der Energieproduktion kommen. Diese muss das EMS entsprechend durch regelbare Kraftwerke, sowie durch Aus- bzw. Einspeisung von Energie aus bzw. in Speicherkraftwerke ausgleichen.

Im Folgenden wird eine zentrale Steuerung beispielhaft durch die im Projekt ”Das regenerative Kombikraftwerk” (Vorgängerprojekt des in Kap. 1.5 beschriebenen Projektes ”Kombikraftwerk 2”) beschrieben. In diesem wurde ein Großteil der Energie durch PV und Windenergie erzeugt. Weiterhin kommt ein Pumpspeicherkraftwerk zur Speicherung von Energie, sowie von Biogas-Blockheizkraftwerke (BHKWs) zum Einsatz. Es gibt dabei sowohl solche Biogas-BHKWs, die innerhalb kürzester Zeit regelbar sind, als auch solche, die eine mittelfristige Planung benötigen.
In Abhängigkeit von erwarteter Auslastung, Wetterprognosen durch den Deutsche Wetterdienst, sowie Speicherständen von Biogas-BHKWs und Pumpspeicherkraftwerken wird ein Fahrplan, welcher die Regelung der Kraftwerke übernimmt, erstellt. Dieser wird stündlich neu berechnet und deckt jeweils die folgenden 48 Stunden ab. Biogas-BHKWs, die eine mittelfristige Planung benötigen, werden dabei zum Ausgleich von erwarteten Schwankungen (z.B. durch vorausgesagte Wolkendecken oder Windflauten) genutzt. Dagegen können Biogas-BHKWs, welche innerhalb kürzester Zeit regelbar sind, unerwartete Schwankungen (z.B. unerwartete Wettervorkomm-

nisse) ausgleichen. Des Weiteren können Schwankungen durch das Pumpspeicherkraftwerk und notfalls auch durch Ex- bzw. Import aus dem europäischen Ausland erfolgen. Falls die Energieproduktion so hoch ist, dass dieses nicht ausgeglichen werden kann, kann außerdem auch PV und Windenergie heruntergefahren werden (z.B. durch ändern der Rotorblätter-Winkel von Windkraftanlagen).

Die Erzeuger senden durchgehend Daten, in denen die tatsächliche Einspeisung der prognostizierten gegenübergestellt wird, an das EMS. Es werden dann minütig entsprechende Anpassungen durchgeführt.

1.4.2 Dezentrale Steuerung

Dimeas und Hatziargyriou [3] stellen eine weitere Möglichkeit um VKs zu steuern vor. Dabei wird mithilfe eines Multi Agenten System (MAS) ein dezentralisierter Ansatz verfolgt. Das verwendete MAS unterteilt Agenten in die drei Ebenen: die Field-Ebene, die Management-Ebene und die Enterprise-Ebene.

Ein Agent der Field-Ebene steuert einen Energieerzeuger oder eine regelbare Last. Dieser Agent kennt zunächst nur Informationen über die eigene Anlage und ggf. über lokale Einflüsse, wie z.B. die Umgebungstemperatur. Durch Kommunikation mit seinen Nachbarn kann der Agent auch weitere Informationen austauschen.

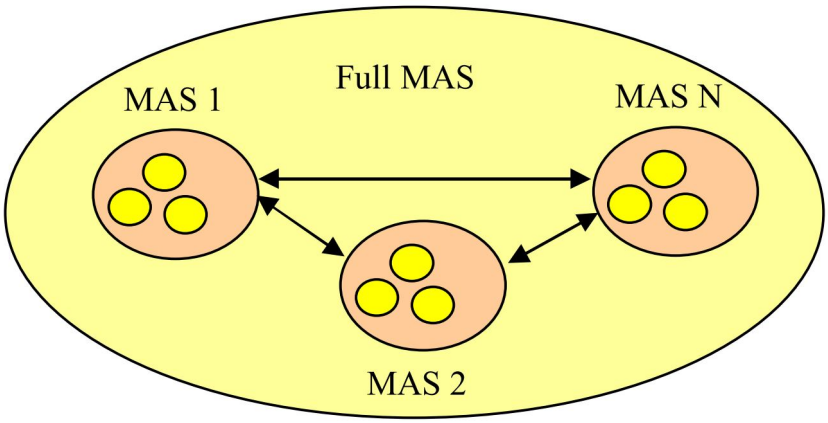


Abb. 1.1: Aufbau des MAS [3]

Diese Agenten werden in geographische Gebiete unterteilt und bilden somit wiederum Unter-MAS (MAS 1 bis MAS N in Abb. 1.1). Die Kommunikation zwischen den Unter-MAS werden von Agenten der Management-Ebene übernommen. Jedes Unter-MAS hat außerdem auch einen Agenten der Enterprise-Ebene, welcher sich um die Vermarktung der Energie an dem Energiemarkt kümmert. Dabei müssen die Aufgaben der Enterprise- und Management-Ebene nicht zwangsläufig von eigenständigen Agenten übernommen werden, sondern dieses kann auch durch einen Field-Agenten durchgeführt werden. Dieses macht das System ausfallsicherer, da diese Aufgaben von anderen Agenten übernommen werden können.

Agenten beeinflussen sich dabei durch Aktionen gegenseitig. Entschließt sich beispielsweise ein Agent, der einen Energiespeicher steuert dazu, Energie in das System einzuspeisen, beeinflusst das die Umgebung und andere Agenten.

Da in diesem System nicht alle Informationen an eine Leitstelle gesendet werden müssen, müssen weniger Informationen über das Kommunikationsnetz versendet werden. Trotzdem werden mehr Informationen mit einbezogen, als in einer zentralen Steuerung. So kann z.B. ein Agent der eine Kraft-Wärme-Kopplung Anlage steuert, lokale Informationen wie Umgebungstemperatur sowie Wärmebedarf mit in Entscheidungen einfließen lassen.

1.5 Kombikraftwerk 2

In diesem Kapitel wird das deutsche Projekt Kombikraftwerk 2 vorgestellt, in dem ein Modell zur Stromversorgung von Deutschland ausschließlich durch EE entwickelt wurde. Der Begriff Kombikraftwerk ist hierbei als Synonym zu dem in dieser Ausarbeitung verwendeten Begriff VK zu verstehen. Das Projekt lief über einen Zeitraum von etwa drei Jahren und endete im Dezember 2013. Das gesamte Kapitel beruht auf dem Abschlussbericht, sowie dem Kurzbericht des Projektes [7, 8].

In dem Vorgängerprojekt regeneratives Kombikraftwerk (RKW, siehe auch Kap 1.4), welches im Herbst 2007 Abgeschlossen wurde, wurde bereits gezeigt, dass die Energieproduktion ausschließlich durch EE für ganz Deutschland möglich ist. Im Kombikraftwerk 2 wurde darauf eingegangen, ob auch die Netzdienstleistungen erfüllt werden können. Als relevante Netzdienstleistung sind vor allem die Frequenz- und die Spannungshaltung zu nennen. Eine zu große Abweichung von Frequenz oder Spannung, kann zu Stromausfällen oder sogar zur Beschädigung von elektrischen Geräten führen. Aktuell werden diese Netzdienstleistungen hauptsächlich durch Kohle-, Erdgas- und Atomkraftwerke bereitgestellt. Das Hauptziel war es zu überprüfen ob und wie dieses auch mit ausschließlich EE-Anlagen in Zukunft möglich ist. In dieser Ausarbeitung wird dabei jedoch hauptsächlich auf die technischen Möglichkeiten von EE-Anlagen zur Erbringung von Regelenergie, welche zur Frequenzhaltung benötigt wird, eingegangen (siehe Kap. 1.5.1).

Zunächst wurde ein konsistentes Zukunftsszenario entwickelt. Das Szenario hatte eine hohe räumliche Auflösung, also exakte örtliche Angaben für Stromerzeuger, Speicherkraftwerke, Verbraucher und Stromnetz. Das Szenario ging über ein Jahr mit einer zeitlichen Auflösung von einer Stunde.

Wichtig war es zunächst abzuschätzen, wie hoch der Bedarf für elektrische Energie in der Zukunft sein wird. Die Forscher gingen davon aus, dass zwar die Anzahl der Geräte die elektrische Energie benötigen zwar deutlich zunehmen, diese aber auch deutlich effizienter seien als heutzutage. Dementsprechend wurde davon ausgegangen, dass dieses sich etwa ausgleicht und der Energiebedarf deshalb mit dem Heutigen zu vergleichen ist.

Als EE wurde ein Strommix von Wind (60%), PV (20%), Bioenergie (10%) sowie Wasserkraft und Geothermie (10%) verwendet. Die Stromerzeugung wurde mit Hilfe der Wetterdaten aus dem Jahr 2007 simuliert. Da eine ausreichende Energieversorgung auch in Zeiten von wenig Wind und Sonne gewährleistet werden muss, werden außerdem Speicherkraftwerke benötigt. Dazu wurden hauptsächlich Power-To-Gas Kraftwerke verwendet, welche eine Energie-Speicherung über große Zeiträume ermöglichen. Bei den verwendeten Power-To-Gas Kraftwerken wird durch elektrischer Energie ein Methan-Gemisch aus Wasser und CO2 gewonnen, welches in Unterirdischen Tanks gespeichert werden kann. Zu einem späteren Zeitpunkt kann das Gas dann wieder in elektrische Energie zurück gewandelt werden. So wird eine Energiespeicherung über mehrere Monate problemlos möglich.

1.5.1 Frequenzhaltung

Damit die Frequenz konstant bleibt, müssen sich Stromverbrauch und Erzeugung ausgleichen. Das bedeutet, die Erzeugung muss an den Verbrauch angepasst werden. Auf Abweichungen muss dabei sehr schnell reagiert werden. Dafür werden sogenannte Regelkraftwerke benötigt, welche die Fähigkeit haben die Energieproduktion schnell zu erhöhen, bzw. zu verringern. Ist der Verbrauch nun zu hoch oder zu niedrig, wird die Energieproduktion entsprechend hoch- bzw. heruntergefahren. Zu beachten ist auch, dass Energie Produktion von Windkraft und PV-Anlagen durch Schwankungen von Windstärke und Sonneneinstrahlung ebenfalls starken Schwankungen unterliegen kann. Somit spielt die Regelenergie eine sogar noch größere Rolle in diesem Zukunftsszenario.

Als Regelkraftwerke werden klassischerweise zumeist Gaskraftwerke oder Pumpspeicherkraftwerke eingesetzt. Die Forscher von Kombikraftwerk 2 haben jedoch untersucht, ob dieses auch mit EE-Anlagen möglich ist. Dabei wurde in einem Feldtest überprüft, ob die Anlagen die technischen Voraussetzungen haben, also ob diese die Produktion schnell genug hoch- bzw. herunterfahren können. Es wurde ein VK aus PV-Anlagen, Windparks und Biogasanlagen vernetzt um zu überprüfen, ob die Anfor-

derungen erfüllt werden können. In Abb. 1.2 sind alle Anlagen, die an dem Feldtest teilgenommen haben inklusive der maximalen Leistung aufgelistet, der Großteil der Leistung wird dabei von den Windparks zur Verfügung gestellt. Es ist jedoch zu beachten, dass diese Erzeuger Zusammensetzung nicht repräsentativ ist, sondern vielmehr durch die Verfügbarkeit von nutzbaren Anlagen bestimmt wurde. Deshalb entspricht die Zusammensetzung auch nicht dem in Kap. 1.5 vorgestellten Strommix.

Anlage(n)	Standort	Anzahl Anlagen	Leistung [MW]
Windpark Altes Lager der ENERCON GmbH	Brandenburg	18	37,2
Windpark Feldheim der energiequelle GmbH	Brandenburg	19	39,2
PV-Anlagen auf Privatgebäuden	Raum Kassel	12	knapp 1,0
PV-Großanlagen	Raum Kassel	3	
Biogasanlage Wallerstädten	Hessen	1	1,2
Biogasanlage Mittelstrimming	Rheinland-Pfalz	2	0,5
Biogasanlage Zemmer	Rheinland-Pfalz	2	1,4
Biogasanlage Heilbachhof	Rheinland-Pfalz	2	0,5

Abb. 1.2: Anlagen die zu einem virtuellen Kraftwerk vernetzt wurden, um Regelenergie zur Verfügung zu stellen [8]

Der Feldtest setzte sich aus drei Phasen zusammen. In Phase eins und drei, wurden feste Sollwerte angegeben, die das VK erreichen musste. In Phase zwei hingegen, orientierte sich der Sollwert an der Netzfrequenz. Die Ergebnisse der drei Phasen sind in Abb. 1.3 zu sehen.

Laut den Forschern vom Kombikraftwerk 2, sind EE-Anlagen demnach technisch bereits heute in der Lage die benötigten Systemdienstleistungen bereit zu stellen. Für Windenergie und PV-Anlagen ist dieses jedoch nur dann möglich, wenn ausreichend Wind weht, bzw. Sonne scheint.

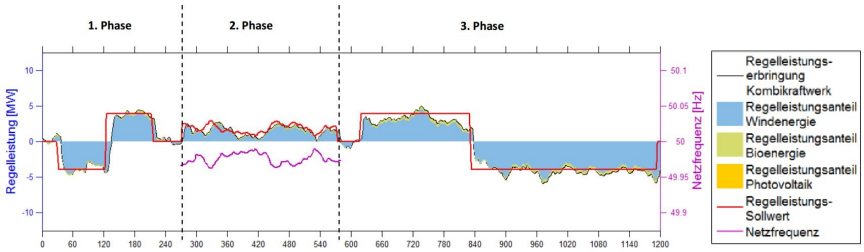


Abb. 1.3: Ergebnisse der drei Phasen im Feldversuch [8]

1.5.2 Ergebnisse

Die Ergebnisse des Projekts Kombikraftwerk 2 legen nahe, dass eine stabile und sichere Energieversorgung auf Grundlage von 100% EE bereits heute technisch machbar ist. Ausfälle und Abweichungen können durch den Verbund eines VK besser kompensiert werden. Benötigte Systemdienstleistungen können prinzipiell durchaus von EE-Anlagen übernommen werden. Dafür muss jedoch die Markt- und Systemintegration noch angepasst werden. Es müssen außerdem leistungsfähige und sichere Kommunikationsstandards zum Überwachen und Steuern von den Anlagen festgelegt werden.

1.6 Next Pool

In diesem Kapitel wird das VK Next Pool von der Next Kraftwerke GmbH beschrieben. Bei Next Pool handelt es sich um ein kommerzielles VK, in welches verschiedenste Anlagentypen integriert werden. Unter anderem werden PV, Windenergie, Biogasanlagen und BHKW integriert. Next Pool besteht mittlerweile aus 25.786 Anlagen mit einer maximalen Leistung von 1.539 MW (Stand Feb. 2015) [1] und gehört damit zu den größten VKs in Europa. Mit dieser Leistung kann Next Pool durchaus mit leistungsfähigen aktuellen Kernkraftwerken mithalten.

Mit dem Fernwirksystem ”Next-Box” werden die Anlagen vernetzt und bieten somit eine Schnittstelle zwischen Anlagen und EMS. Dadurch wird eine zentrale Überwachung jeder Anlage, sowie eine Regelung der regelbaren Anlagen durch ein EMS möglich. Die Kommunikation findet über eine getunnelte Verbindung per GPRS-Mobilfunk statt [2].

Wetterprognosen werden eingesetzt um zu einer optimalen Vermarktung von Windenergie und PV beizutragen. Diese EE werden dann an der Strombörse direktvermarktet. Auch Regelenergie wird vom Next Pool verkauft, wobei insbesondere auf die Energie von Biogasanlagen zurückgegriffen wird. Der Einsatz dieser wird dabei optimiert, so dass elektrische Energie möglichst dann erzeugt wird, wenn es zu hohen Preisen verkauft werden kann. Auch können die Anlagen heruntergefahren werden, um negative Regelenergie zur Verfügung zu stellen. Durch die geringen Leistungen der Anlagen kann durch Wegschalten bzw. Zuschalten einzelner Anlagen eine hohe Flexibilität erreicht werden.

Dem Anlagenbetreiber werden dabei nicht nur die zur Verfügung gestellte Leistung bezahlt, sondern auch die Regelung der Anlage (Arbeitspreis) vergütet [2]. Bei einem Abruf von Regelenergie, wählt ein Algorithmus in dem EMS eine oder mehrere Anlage aus und regelt diese entsprechend. Der Anlagenbetreiber wird von dem Abruf via SMS informiert.

1.7 Fazit

Das Konzept des Virtuellen Kraftwerks (VK) ist ein sinnvoller Ansatz um Kleinerzeuger von elektrischer Energie zu einem großen Kraftwerk zusammenzuschließen. Dadurch können Leistungsschwankungen, die gerade durch den Einsatz von Wetterabhängigen Erzeugern, wie Windkraft und Photovoltaik-Anlagen häufig auftreten können, leichter ausgeglichen werden. Außerdem kommen durch den Zusammenschluss größere Leistungen zusammen, wodurch auch ein Einsatz an der Strombörse möglich wird.

Zur Steuerung eines VK können verschiedene Ansätze verwendet werden. In dieser Ausarbeitung wurden ein zentraler und ein dezentraler Ansatz vorgestellt. Der dezentrale Ansatz, welcher durch ein Multi-Agenten System verwirklicht wurde, bietet dabei einige Vorteile gegenüber dem zentralen Ansatz.

Das Forschungsprojekt Kombikraftwerk 2 zeigt dabei, wie ein VK auch Netzdienstleistungen wie z.B. Regelenergie zur Verfügung stellen kann. Das kommerzielle VK Next Pool zeigt wie bereits Heute große Energiemengen durch Bündelung von kleinen Anlagen, sehr erfolgreich sein können.

Literaturverzeichnis

[1] Zuletzt besucht: [29.05.2015]. URL: https://www.next-kraftwerke.de/unternehmen.

[2] Zuletzt besucht: [29.05.2015]. URL: https://www.next-kraftwerke.de/energie-blog/stromhandel-regelenergie.

[3] AL Dimeas und ND Hatziargyriou. „Agent based control of virtual power plants“. In: *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*. IEEE. 2007, S. 1–6.

[4] *Gesetz für den Ausbau erneuerbare Energien (Erneuerbare-Energien-Gesetz - EEG 2014), Nicht-amtliche Lesefassung des EEG in der ab 1. August 2014 geltenden Fassung*.

[5] Nikos Hatziargyriou, Hrsg. *Microgrids Architectures and Control*. John Wiley & Sons, 2014.

[6] Martin Kaltschmitt, Wolfgang Streicher und Andreas Wiese. *Erneuerbare Energien*. Springer, 2006.

[7] Kaspar Knorr u. a. *Kombikraftwerk 2 Abschlussbericht*.

[8] Kaspar Knorr u. a. *Kombikraftwerk 2 Kurzbericht*.

[9] P Lombardi, M Powalko und K Rudion. „Optimal operation of a virtual power plant“. In: *Power & Energy Society General Meeting, 2009. PES'09. IEEE*. IEEE. 2009, S. 1–6.

Kapitel 2
Mikro Blockheizkraftwerke

Carsten Krüger

Zusammenfassung Große thermische Kraftwerke erzeugen im Normalfall den Strom für Ein- oder Mehrfamilienhäuser und geben die dabei erzeugte Hitze an die Umwelt ab. Eine Innovative Methode zur eigenen Energieversorgung ist die Kraft-Wärme-Kopplung unter der Verwendung von Blockheizkraftwerken, die sowohl die elektrische als auch die thermische Leistung bereitstellt. BHKW-Anlagen sollen im Rahmen einer Projektgruppe als dezentrale Energieversorger zu einem virtuellen Kraftwerk zusammengeschlossen werden und damit die Möglichkeit bieten besser am Energiemarkt zu handeln. Im ersten Abschnitt wird auf die Betriebs- und Funktionsweise einer BHKW-Anlage eingegangen. Anschließend werden verschiedene Kostenarten betrachtet und Informationen über bestehende Förderungen hervorgehoben.

2.1 Einleitung

Die Energieversorgung von Ein- oder Mehrfamilienhäusern sowie Wohnkomplexen erfolgt in der Regel über eine getrennte Erzeugung der Strom- und Wärmeenergie. Ein Großteil der benötigte elektrische Leistung (Strom) wird durch große thermische Kraftwerke unter dem Einsatz von Steinkohle, Braunkohle und Kernenergie erzeugt. In dem Energieerzeugungsprozess entstehen hohe Wärmeverluste, da die thermische Leistung (Wärme) ungenutzt an die Umgebung abgeführt wird. Zusätzlich entstehen Übertragungsverluste bei der Übertragung der elektrischen Leistung zum Endverbraucher. Die benötigte Heizwärme für den Verbraucher wird durch Kesselanlagen unter dem Einsatz fossiler Brennstoffe autark erzeugt. In Anbetracht der hohen Wärmeverluste der thermischen Kraftwerke, ergibt sich insgesamt eine geringe Energieeffizienz. Ein innovatives Verfahren zur eigenen Erzeugung von thermischer und elektrischer Leistung ist die Kraft-Wärme-Kopplung (KWK) unter der

Carl von Ossietzky Universität Oldenburg
E-mail: carsten.krueger@uni-oldenburg.de

Verwendung von Mikro-Blockheizkraftwerken (BHKW). Dieses Verfahren bietet eine bessere Energieeffizienz und einen geringeren Schadstoffausstoß. Im Zusammenhang mit der Projektgruppe virtuelles Kraftwerk können Mikro-BHKW als dezentrale Energiequelle eingesetzt werden. Dieses Kapitel befasst sich zunächst mit dem Thema der Kraft-Wärme-Kopplung im Bezug auf Blockheizkraftwerken und deren verschiedenen Betriebsweisen. Anschließend erfolgt ein kurzer Einblick in die Emissionsgesetze, Richtlinien und Fördergelder sowie wichtige Aspekte der Kostenplanung. Abschließend wird in einem Fazit auf eine mögliche Verwendung von Mikro-BHKWs im Rahmen der Projektgruppe eingegangen.

2.2 Kraft-Wärme-Kopplung

Unter der Kraft-Wärme-Kopplung wird die gleichzeitige Erzeugung von mechanischer und nutzbarer thermischer Energie verstanden. Die mechanische Energie wird häufig in einem Generator in elektrische Energie umgewandelt. Zur Umsetzung dieses Verfahrens wird eine Wärmekraftmaschine, ein Generator und ein Wärmekreislauf zur Auskopplung der Abgas- und Kühlwasserwärme benötigt.[9]

2.2.1 Blockheizkraftwerke

Blockheizkraftwerke (BHKW) verwenden das Prinzip der KWK und bestehen aus den benötigten Komponenten der KWK. In Abbildung 2.1 ist ein möglicher Kreislauf zur Strom- und Wärmeerzeugung durch ein BHKW dargestellt. Blockheizkraftwerke unterscheiden sich vor allem in der Leistungsstärke und der verwendeten Wärmekraftmaschine.

2.2.1.1 BHKW Klassifikation

Die Blockheizkraftwerke lassen sich anhand ihrer elektrischen Leistung in Kilowatt (kW) klassifizieren. Diese Leistungsgruppen sind nicht genormt, sodass Unterschiede in der Granularität zwischen den verschiedenen Instanzen vorliegen. Die Einteilung in Nano-, Mikro-, Mini-, und (Groß-) BHKW wird häufig von verschiedenen Händlern vorgenommen. Nach dem Bundesverband für KWK wird von Nano-BHKW bei einer elektrischen Leistung von weniger als 1 kW, von Mikro-BHKW bei weniger als 15 kW und von Mini-BHKW bei weniger als 50 kW gesprochen. Vergleichsweise ist die Klassifizierung der KWK Richtlinien des Europäischen Parlaments deutlich gröber und richtet sich nach den verschiedenen Fördergeldgruppen. Alle Anlagen mit einer elektrischen Leistung von weniger als 50 kW werden als Kleinstanlagen eingestuft.[11]

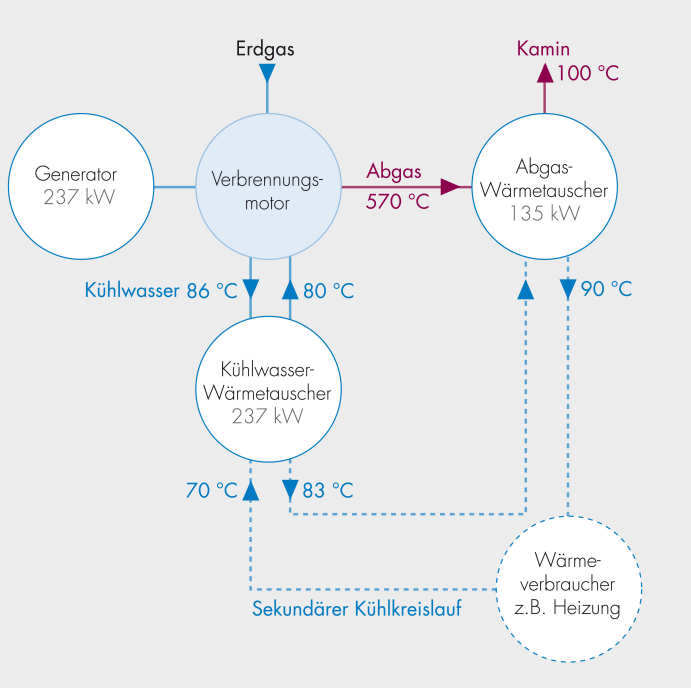


Abb. 2.1: Auskopplung der Abgas- und Kühlwasserwärme in einem BHKW[4]

2.2.1.2 BHKW Wärmekraftmaschinen

Innerhalb eines Blockheizkraftwerk wird eine Wärmekraftmaschine benötigt, um aus fossilen Rohstoffen elektrische und thermische Leistung zu erzeugen. Im Nachfolgendem wird kurz auf die verschiedenen Typen von Wärmekraftmaschinen eingegangen. [6]

- Verbrennungsmotor: Bei einem Verbrennungsmotor wird ein Gas-Luftgemisch im Zylinder verdichtet und durch einen Zündfunken zum Explodieren gebracht. Durch den Explosionsdruck wird der Kolben betrieben und mit einem Generator, die dabei entstandene Kraft, in elektrische Leistung umgewandelt. Zusätzlich entstehen Abgase mit hohen Temperaturen und erhitztes Kühlwasser, welches über einen Wärmetauscher in thermische Leistung umgewandelt wird.
- Stirling Motor: Diese Motorvariante besitzt einen Expansionszylinder und einen Kompressionszylinder. Der Expansionszylinder wird extern erhitzt und lässt ein Helium-Gemisch im Zylinder expandieren (ausdehnen). Durch die Ausdehnung des Expansionszylinders wird der Kompressionszylinder sowie das Helium-Gemisch verdichten und bewirkt eine Rotation der Kurbelwelle. Die Kurbelwelle erzeugt anschließend über einen Generator die elektrische Leistung. Die Umwandlung der thermischen Energie erfolgt wie bei den Verbrennungsmotoren.

- Gasturbinen: Gasturbinen bestehen aus einer Gasexpansionsturbine, einem Verdichter und einer Brennkammer. Durch den Verdichter wird die Luft verdichtet und in der Gaskammer mit einem Treibstoff oder Gas vermischt. Anschließend wird dieses Gemisch entzündet und treibt die Gasexpansionsturbine an, wodurch thermische Energie durch die Abgase und elektrische Leistung über eine angetriebene Welle mit Generator erzeugt wird.
- Dampfturbine: Durch einen Dampfkessel wird unter hohem Druck und Wärme Dampf erzeugt. Dieser Dampf treibt eine Turbine an und erzeugt die benötigte Kraft um einen Generator anzutreiben. Die verbleibende Hitze kann über einen Wärmetauscher als thermische Energie genutzt werden.[8]

2.2.1.3 Technische Nebenbedingungen

Einige Einschränkungen bei der Benutzung von BHKW-Anlagen müssen als Nebenbedingungen betrachtet werden. Im Nachfolgenden werden einige Einschränkungen genauer beschrieben.

- Mindestbetriebszeit: Die BHKW-Anlage darf solange nicht heruntergefahren werden, bis die Mindestbetriebszeit erreicht wurde.
- Mindestwartezeit: Nachdem die Anlage abgeschaltet wurde, darf sie erst nach der Mindestwartezeit wieder hochgefahren werden.
- Verzugszeit: Durch die Verzugszeit wird die benötigte Zeit nach dem Start der Anlage angegeben, bis die erste elektrische Leistung erzeugt wird.
- Zeitspanne bis zur maximalen el/th Leistung: Dieser Wert gibt die Dauer bis zur Volllast der BHKW-Anlage für die elektrische und thermische Leistung an.
- Starthäufigkeiten: Der Verschleiß der BHKW-Anlagen durch häufige Starts wird zusätzlich zu der Mindestwartezeit angegeben. Diese Einschränkung bezieht sich zumeist auf die Starts innerhalb eines Jahres.

Genauere Kennzahlen der einzelnen Einschränkungen lassen sich nicht verallgemeinern und hängen sehr stark von der Wärmekraftmaschine und dem Hersteller ab. Zum Beispiel hat eine BHKW-Anlage mit Verbrennungsmotor eine sehr geringe Verzugszeit von wenigen Sekunden, hingegen benötigt eine Gasturbine mehrere Minuten und muss vorab durch Strom angetrieben werden. In Abbildung 2.2 sind die Einschränkungen für eine BHKW-Anlage mit Verbrennungsmotor beispielhaft aufgeführt.

2.3 Betriebsweise der BHKW-Anlagen

Der wirtschaftliche Nutzen einer BHKW-Anlage ist größtenteils abhängig von ihrer Laufzeit unter Volllast. Um möglichst viele Volllaststunden pro Jahr zu erreichen, ist eine Abstimmung auf den Strom- und Wärmebedarf der Abnehmer erforderlich. Ebenso gibt es bei der Wahl von variablen Tarifzonen verschiedene Preiszonen für

Mini-BHKW-Aggregat		
Elektrische Nennleistung P_{el}	[kW]	5
Thermische Nennleistung P_{th}	[kW]	13
Brennstoffleistung	[kW]	20
Zeitspanne bis 90 % P_{el}	[min]	2
Zeitspanne bis 90 % P_{th}	[min]	8
Verzugszeit [1])	[min]	0
Mindestbetriebszeit	[min]	60
Mindestwartezeit	[min]	30

Abb. 2.2: Technische Einschränkungen einer BHKW-Anlage mit Verbrennungsmotor [12]

den Bezug von Strom. Diese bieten die Möglichkeit Strom zu gewissen Tageszeiten teuer oder günstig zu kaufen oder verkaufen. Durch sogenannte Steuer- und Regelgeräte lassen sich die BHKW-Anlagen je nach Betriebsweise konfigurieren. Dabei wird unter Wärme-geführte, Strom-geführte oder kombinierter Betriebsweise der BHKW-Anlagen unterschieden.[6]

2.3.1 Wärme-geführte Betriebsweise

Bei dieser Betriebsweise erfolgt die Steuerung über die benötigte thermische Leistung der angeschlossenen Verbraucher, wie beispielsweise Heizungsanlagen oder Warmwasser. Zusätzlich werden Informationsquellen wie die Außentemperatur benötigt. Die Leistungsgröße einer BHKW-Anlage wird anhand einer Grundlast berechnet und ergibt sich aus einer möglichst hohen Anzahl an Volllaststunden pro Jahr. Um kurzfristige Schwankungen der Wärmegrundlast zu kompensieren und einen anhaltenden Betrieb zu gewährleisten können weitere Maßnahmen wie Drosselung der Motorleistung und die Verwendung eines Pufferspeichers vorgenommen werden. Darüber hinaus benötigte thermische Leistung wird über einen zusätzlichen Spitzenlastkessel erzeugt. Bei der getrennten Strom- und Wärmeerzeugung wird ein Heizkessel zur Erzeugung der thermischen Leistung benötigt, dieser wird bei der Nutzung einer BHKW-Anlage durch ein Spitzenlastkessel ersetzt. Die generierte elektrische Leistung durch die BHKW-Anlage kann zur Eigenbedarfsdeckung genutzt oder wird für eine Vergütung in das öffentliche Netz eingespeist werden.

2.3.2 Strom-geführte Betriebsweise

Die Strom-geführte Betriebsweise verwendet ein prognostiziertes Lastprofil der elektrischen Leistung von Gebäuden als Kennzahlen, um die Lastregelung der BHKW-Anlage zu steuern. Zusätzlich benötigte elektrische Leistung kann über das öffentliche Netz bezogen werden. Dabei kann insbesondere der Bezug von Strompreisspitzen durch die BHKW-Anlage vermieden werden. Die erzeugte thermische Leistung muss entweder durch den Eigenbedarf, eines Nahwärmenetz oder über ein Fernwärmenetz abgeführt werden. Auch bei der Strom-geführten Betriebsweise ist es möglich die Motorleistung zu Drosseln und einen Pufferspeicher für eine erhöhte Flexibilität zu nutzen.

2.3.3 Wärme/Strom-geführte Betriebsweise

Eine kombinierte Betriebsweise kann unter Umständen zu erhöhten Vollbenutzungsstunden führen. Dabei wird zunächst die Wärmeanforderung als erste Führungsgröße verwendet und nur bei erhöhter Anforderung der elektrischen Leistung wird auf eine Strom-geführte Betriebsweise gewechselt. Die überschüssige Wärmeerzeugung während der Strom-geführten Betriebsweise kann in einem Pufferspeicher zwischengespeichert werden. Bei dieser Betriebsweise ist ein ausschlaggebendes Kriterium die variierenden Strombezugskosten zu unterschiedlichen Tageszeiten.

2.4 Bedarfsdeckung

Bevor eine BHKW-Anlage integriert werden kann, muss die Leistungsstärke festgelegt werden. Die Leistungsstärke berechnet sich aus dem Bedarf an thermischer oder elektrischer Leistung und kann durch eine Bedarfsdeckung repräsentiert werden. Die Bedarfsdeckung unterteilt sich in die Wärme- und Strombedarfsdeckung und repräsentiert dabei die jeweilige Betriebsweise in der die BHKW-Anlage geführt werden soll.

2.4.1 Wärmebedarfsdeckung

Ein genauer Verbrauchsplan über die benötigte thermische Leistung bietet eine gute Grundlage zur Erstellung einer Wärmebedarfsdeckung. Die Leistungsstärke einer BHKW-Anlage ergibt sich aus einer möglichst hohen Anzahl an Vollbenutzungsstunden pro Jahr und einer möglichst großen Abdeckung der benötigten thermische Leistung. Die zusätzlich benötigte thermische Leistung muss über eine Kesselanlage erzeugt werden. Die Abbildung 2.3 zeigt eine prognostizierten Verlauf einer Wär-

mebedarfsdeckung eines Jahres. Der thermische Leistungsbedarf wird absteigend geordnet als Jahresdauerlinie dargestellt und als Referenzwert für die Leistungsstärke der BHKW-Anlage verwendet.

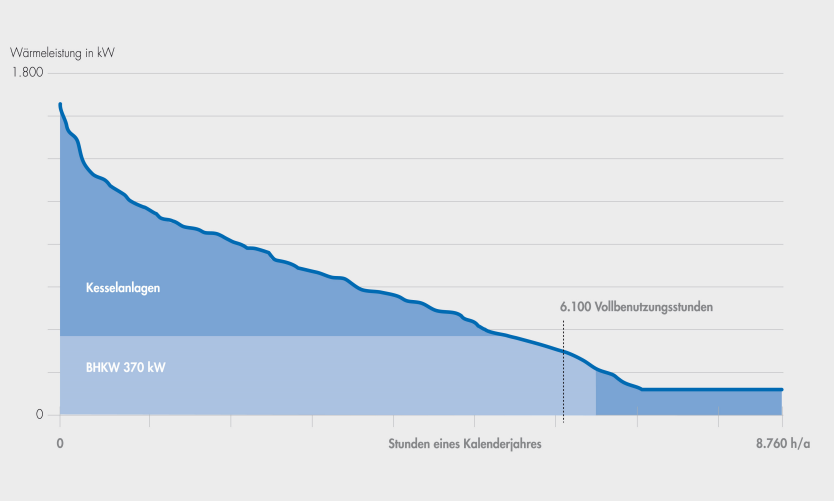


Abb. 2.3: Jahresdauerlinie der Wärmebedarfsdeckung durch BHKW und Kesselanlage[1]

2.4.2 Strombedarfsdeckung

Die benötigte elektrische Leistung variiert stark innerhalb eines Tages und bleibt über das Jahr größtenteils Konstant. Zusätzlich kann die zu viel erzeugte elektrische Leistung in das öffentliche Netz eingespeist werden. In Abbildung 2.4 ist eine Prognostizierte Tagesauslastung grafisch dargestellt. Ausgehend von einer Abnahme der thermischen Leistung kann die Leistungsstärke der BHKW-Anlage im Bezug auf der Strombedarfsdeckung stark variieren und auch eine autarke Stromversorgung zulassen.

2.5 Modularer Aufbau

Eine Reduzierung der elektrischen und thermischen Leistung ist zumeist nur beschränkt möglich, da die meisten Hersteller eine Drosselung der BHKW-Anlagen nur bis auf 60 Prozent der Maximallast vorgeben. Eine stärkere Drosselung der

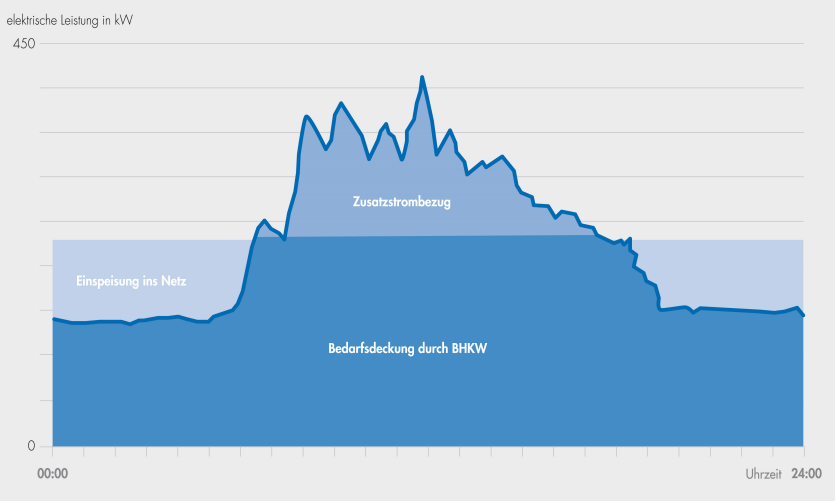


Abb. 2.4: Tagesauslastung der Strombedarfs, Bedarfsdeckung durch BHKW[2]

BHKW-Anlage würde die Energieeffizienz extrem verringern. Um eine erhöhte Flexibilität der Anlage zu erhalten kann ein modularer Aufbau mit mehreren Aggregaten eingesetzt werden. Jedes Aggregat besteht aus einer eigenen Wärmekraftmaschine und einer separaten Kühlung. Diese lassen sich einzeln aktivieren und in den Kreislauf der BHKW-Anlage integrieren. Dabei sollte beachtet werden, dass bei einer BHKW-Anlage mit zwei Aggregaten höhere Kapital gebundene Kosten auftreten als bei einer BHKW-Anlage mit nur einem Aggregat. Des Weiteren gibt es folgende Vor- und Nachteile.

V: Stufenweise Anpassung der Leistung
V: Höhere Verfügbarkeit und Ausfalltoleranz
V: Erweiterbarkeit der BHKW-Anlage um weitere Aggregate zur Erhöhung der Gesamtleistung
N: Erhöhte Kosten für Wartung und Instandhaltung
N: Höhere Anschaffungskosten
N: Der elektrische Wirkungsgrad kleiner Anlagen ist geringer als der Wirkungsgrad großer Anlagen (Abb. 2.5)

2.6 Richtlinien

Im Rahmen des Bundesimmissionsschutzgesetz (BImSchG) zum Schutz der Bevölkerung vor schädlichen Belastungen durch Luftverunreinigungen, Geräuschen und anderen Einflüssen, unterliegt der Betrieb von Anlagen gewissen Auflagen. Die

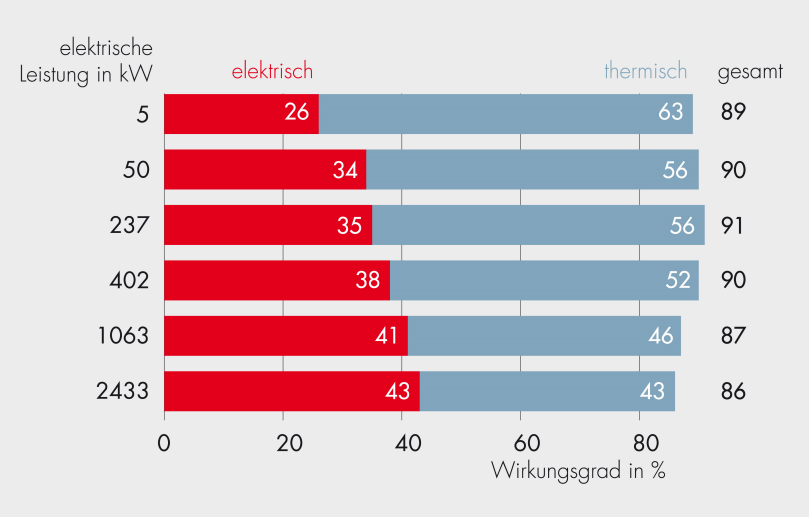


Abb. 2.5: Wirkungsgrad von BHKW-Anlagen[3]

Richtlinien unterteilen sich in die genehmigungsbedürftigen (§4-§12) und nicht genehmigungsbedürftigen (§22-§25) Anlagen. Zusätzlich gibt es die 4. Bundesimmissionsschutzverordnung (BLmSchV) in der spezifiziert wird, dass Verbrennungsanlagen und Gasturbinenanlagen mit mehr als 1MW erzeugter elektrischer Leistung als genehmigungsbedürftig einzustufen sind. Dementsprechend sind alle darunter nicht genehmigungsbedürftig und benötigen keine Abnahme der TA-Luft Vorgaben. Dennoch wird in §22 festgehalten, dass alle Anlagen den möglichen Stand der Technik zur Verringerung des Schadstoffausstoßes erbringen oder auf ein Minimum begrenzen müssen. Zusätzlich müssen die Richtlinien für die TA-Lärm auch für die nicht genehmigungsbedürftigen Anlagen eingehalten werden. Die TA-Lärm Bestimmungen beinhalten innerhalb von Gebäuden einen Wert tagsüber von maximal 35 dB und nachts von 25 dB. Diese Werte werden von den meisten Herstellern berücksichtigt und durch verbaute Dämmungen erzielt. [6] [7]

2.7 Pufferspeicher

Abhängig von dem Einsatzzweck und der Betriebsweise einer BHKW-Anlage ist es mehr oder weniger sinnvoll ein Pufferspeicher zu verwenden. Die erzeugte thermische Energie kann in einem Pufferspeicher kurzzeitig zwischengespeichert werden und ermöglicht eine erhöhte Flexibilität der BHKW-Anlage. Einige BHKW-Anlagen haben durch den Hersteller integrierte Pufferspeicher. Diese weisen zumeist eine geringe Kapazität auf und bieten nur eine geringfügig erhöhte Flexibilität. Einen Über-

blick, über die Kapazität von Pufferspeichern und die dazugehörigen Preise, kann in der Abbildung 2.6 entnommen werden. Im Bezug auf die Wärmebedarfsdeckung können mehr Vollbenutzungsstunden erzielt werden, dadurch dass bei geringer Wärmeanforderung die benötigte thermische Leistung nicht über einen Spitzenlastkessel sondern über den Pufferspeicher bereitgestellt wird. Bei der Strombedarfsdeckung kann der Pufferspeicher während erhöhter Anforderung der elektrischen Leistung die erzeugte thermische Leistung aufnehmen und gezielt bei geringer Anforderung der elektrischen Leistung abgeben. Des Weiteren kann die Verwendung eines Pufferspeichers und dadurch verringerte Starts der BHKW-Anlage dafür sorgen, dass die technische Lebensdauer erhöht wird und weniger Wartungs- und Instandhaltungskosten anfallen.[13]

Preisliste

Typ / Bestell-Nr.	Preis EUR zuzügl. Mwst.
PS - 200	518,--
PS - 300	601,--
PS - 500	695,--
PS - 600	737,--
PS - 750	788,--
PS - 950	972,--
PS - 1500	1.402,--
PS - 2000	2.331,--

Abb. 2.6: Preisliste für Pufferspeicher

2.8 Kostenplan

Ein wesentlicher Aspekt der BHKW-Planung ist die Durchführung einer Analyse der Wirtschaftlichkeit unter Betrachtung der Leistungsgröße und Aufbau der Anlage. Ebenso werden die Kosten für die Installation, Instandhaltung und den Betrieb mit einbezogen, sowie Vergütungen durch Förderungen raus gerechnet. Mit eine Referenzmodell, ohne die Verwendung einer BHKW-Anlage, kann eine geeignete Übersicht geschaffen werden um verschiedene Investitionen abzuwägen. Die Kosten für die Anschaffung und den Betrieb einer Anlage lassen sich in 3 Arten unterscheiden. Des Weiteren gibt es Kosten und Vergütungen, die nicht direkt im Zusammenhang mit einer BHKW-Anlage stehen.[5]

2.8.1 Kapital-gebundene Kosten

Diese Kosten resultieren aus den baulichen Komponenten, die initial für die Investition getätigt werden. Also im Fall einer BHKW-Anlage aus dem BHKW-Modul, der Regelungs- und Steuerungseinrichtungen, den Anschlüssen und weitere Komponenten für den Wärme- und Stromkreislauf. Die Kosten werden im Normalfall auf 10-15 Jahre umgelegt und mit einem Zinssatz eines Finanzzeitraums versehen.

2.8.2 Verbrauchs-gebundene Kosten

Zu den Verbrauchs-gebundenen Kosten zählen hauptsächlich die Kosten für den fossilen Brennstoff der jeweiligen BHKW-Anlage. Zusätzlich muss der Verbrauch durch den Spitzenlastkessel betrachtet und ebenfalls als Kosten aufgeführt werden. Bei der Planung werden Referenzwerte sowie Prognosen für die Preisentwicklung mit einbezogen.

2.8.3 Betriebs-gebundene Kosten

Kosten für die Wartung und Instandhaltung, Personalaufwand, Verwaltungsaufwand und sonstige Aufwendungen werden unter Betriebs-gebundene Kosten kategorisiert. Dieser Service wird im Allgemeinen auch vom Hersteller oder Fremdfirmen mit angeboten. Dabei entstehen Kosten pro erzeugter kWh elektrischer Leistung, die abhängig von der erzeugten Menge variieren.

2.8.4 Veränderte Strombezugskosten

Im Normalfall wird die BHKW-Anlage nicht den kompletten Strombedarf decken und muss extern über das öffentliche Netz erworben werden. Durch die verringerte Menge der abgenommenen elektrischen Leistung, kann es zu verschlechterten Kondition seitens der Anbieter kommen. Dieser Aufpreis muss mit in die Kostenrechnung integriert werden.

2.8.5 staatliche Förderungen

Im Rahmen des Klimawandels und dem Versuch die Emissionen in Deutschland zu reduzieren werden BHKW-Anlagen vom Staat gefördert. Die Höhe des Zuschlags geht nach erzeugten kW elektrischer Leistung, wobei der höchste Zuschlag mit 5,41

Cent/kWh für eine Leistungsstärke von maximal 50kW oder für Brennstoffzellen gegeben wird. Die Förderung unterliegt unterschiedlichen Einschränkungen wie zum Beispiel nur für die ersten 30.000 Vollbenutzungsstunden oder für die ersten 10 Jahre. Ebenso ist die Förderung an einer zeitlichen Richtlinie gebunden und wird nur bis ende 2020 gewährt. Weitere Zuschlagsstufen können aus der Abbildung 2.7 entnommen werden.[10]

Elektrische Leistung von KWK-Anlagen bis 50 kW und Brennstoffzellen	Ct/kWh KWK-Strom	Aufnahme des Dauerbetriebes	Maximale Vollbenutzungsstunden
Bis 50 kW und Brennstoffzellen	5,41	19.06.2012 - 31.12.2020	30.000 oder wahlweise 10 Jahre
Elektrische Leistung von KWK-Anlagen > 50 kW			
Für den Leistungsanteil bis 50 kW	5,41	19.06.2012 - 31.12.2020	30.000
Für den Leistungsanteil über 50 kW - 250 kW	4	19.06.2012 - 31.12.2020	30.000
Für den Leistungsanteil über 250 kW - 2 MW	2,4	19.06.2012 - 31.12.2020	30.000
Für den Leistungsanteil über 2 MW	1,8	19.06.2012 - 31.12.2020	30.000
Für den Leistungsanteil über 2 MW	2,1	01.01.2013 - 31.12.2020	30.000 und für Anlagen im Anwendungsbereich des Treibhausgas-Emissionshandelsgesetzes

Abb. 2.7: Auskopplung der Abgas und Kühlwasserwärme in einem BHKW

2.9 Fazit: BHKW-Anlagen in der PG

Im Rahmen der Projektgruppe können die BHKW-Anlagen als sehr sicherer und flexibler Energieerzeuger betrachtet werden, da die Anlagen nicht wie beispielsweise Solaranlagen oder Windkraftwerke von den Wetterverhältnissen abhängig sind. Durch die erhaltene Flexibilität ist es möglich erzeugte elektrische Energie zu geeigneten Zeiten in das Netz einzuspeisen, um bessere Konditionen pro kWh zu erzielen. Trotz der Unabhängigkeit der BHKW-Anlagen gegenüber Wetterverhältnissen müssen einige Einschränken beachtet werden, so ist es zum Beispiel wichtig einen Abnehmer für die erzeugte thermische Leistung zu finden oder die begrenzte Möglichkeit BHKW-Anlagen auf eine geringere Leistung zu drosseln. Durch die Verwendung von Pufferspeichern wird der Freiheitsgrad in der die BHKW-Anlage den Einschränkungen unterliegt vergrößert. Des Weiteren sind die Rahmenbedingungen in denen eine BHKW-Anlage verwendet wird ausschlaggebend für den Kostenplan und für eine erfolgreiche Umsetzung.

Literaturverzeichnis

[1] *ASUE Grafiken Geordnete Jahresdauerlinie des Wärmebedarfs Wärmebedarfsdeckung durch BHKW und Kesselanlagen*. Mai 2015. URL: http://asue.de/themen/blockheizkraftwerke/grafiken/grafik-bhkwgrund-2010-12.html.

[2] *ASUE Grafiken Tageslastgang des Strombedarfs Bedarfsdeckung durch BHKW, Zusatzstrombezug und Einspeisung*. Mai 2015. URL: http://asue.de/themen/blockheizkraftwerke/grafiken/grafik-bhkwgrund-2010-13.html.

[3] *ASUE Grafiken Wirkungsgrade von BHKW mit unterschiedlichen Leistungen*. Mai 2015. URL: http://asue.de/themen/blockheizkraftwerke/grafiken/grafik-bhkwgrund-2010-06.html.

[4] *ASUE Grafiken zur Auskopplung der Abgas und Kühlwasserwaerme in einem BHKW*. Mai 2015. URL: http://asue.de/cms/upload/inhalte/blockheizkraftwerke/grafiken/grafik-bhkwgrund-2010-08_f.jpg.

[5] *BHKW-Fibel Wissen in kompackter Form*. Mai 2015. URL: http://asue.de/cms/upload/broschueren/2012/bhkw_fibel/asue_bhkw%20_fibel_2012.pdf.

[6] *BHKW-Grundlagen*. Mai 2015. URL: http://asue.de/cms/upload/inhalte/blockheizkraftwerke/broschuere/bhkw-grundlagen-2010.pdf.

[7] *Bundesimmisionsschutzverordnung*. Mai 2015. URL: http://www.notstrom-bhkw.de/files/bimschg_verordnungen.pdf.

[8] *Funktionsbetrieb Dampfturbine*. Mai 2015. URL: https://www.energie-lexikon.info/dampfturbine.html.

[9] *Kraft-Waerme-Kopplung, Begriffserklärung*. Mai 2015. URL: http://wirtschaftslexikon.gabler.de/Archiv/58175/kraft-waerme-kopplung-v8.html.

[10] *Kraft-Wärme-Kopplung, Chancen für Wirtschaft und Umwelt*. Mai 2015. URL: http://www.bkwk.de/fileadmin/users/bkwk/aktuelles/Broschur/BKWK_Chance_fuer_Wirtschaft_und%20Umwelt_Broschuere_A4_web.pdf.

[11] *Mikro-KWK und virtuelle Kraftwerke*. Mai 2015. URL: https://www.ffe.de/download/Veroeffentlichungen/2009_Fachtagung_vRoon.pdf.

[12] *Mini-BHKWs noch zu groß für Ein- und Zweifamilienhäuser*. Mai 2015. URL: www.springer-vdi-verlag.de.

[13] *Pufferspeicher, Welcher Speicher passt zu meinem Heizsytem*. Mai 2015. URL: http://www.messerschmid-energiesysteme.de/pufferspeicher.php.

Kapitel 3
EPEX Spot - Produkte

Robert Zilke

Zusammenfassung Diese Ausarbeitung befasst sich mit der Energiebörse EPEX Spot. Das Hauptaugenmerk wird dabei auf die angebotenen Produkte gelegt, welche sich in benutzerdefinierte Blöcke, Standardprodukte und 15 Minuten Blöcke aufteilen lassen. Zusätzlich wird auf die Funktionsweisen der Regelenergie eingegangen. Hierbei werden die Unterschiede zwischen Primär-, Sekundär- und Minutenreserven verdeutlicht.

Carl von Ossietzky Universität Oldenburg
E-mail: Robert.Zilke@uni-oldenburg.de

3.1 Spotmarkt

Ein Spotmarkt ist eine Börse, bei dem der Handel in kurzfristigen Zeiträumen stattfindet. Der Name leitet sich vom englischen „on-the-spot“ ab, was übersetzt soviel bedeutet wie „auf der Stelle“ oder „kurzerhand“. Handelsgeschäfte werden hier in der Regel binnen zwei Tagen abgewickelt, unabhängig von Wochen- oder Feiertagen. In diesem Zeitraum ist der Käufer dazu verpflichtet, den ausgehandelten Betrag an den Verkäufer zu zahlen. Ebenso hat sich der Verkäufer durch den Handel an einem Spotmarkt dazu verpflichtet, die gehandelte Ware im zuvor abgemachten Zeitraum zu liefern. Länger andauernde Handelsgeschäfte werden im abzugrenzenden Terminmarkt getätigt [12]. Am Beispiel des deutschen Terminmarktes für Strom EEX können Stromlieferungen bis zu sechs Jahren vor der Lieferung ausgehandelt werden. Dabei werden verschiedene Produkte mit einer Dauer von wenigen Wochen bis über mehrere Jahre angeboten.

An verschiedenen Spotmärkten werden Wertpapiere, aber auch Edelmetalle und Rohstoffe wie Rohöl und Erdgas gehandelt. Im folgenden wird der Spotmarkt EPEX Spot genauer betrachtet, an dem mit Strom gehandelt wird.

3.2 EPEX Spot

EPEX Spot steht für European Power Exchange und ist ein privates Unternehmen, welches den Stromhandel für die Länder Deutschland, Frankreich, Österreich und Schweiz betreibt. Hierbei handelt es sich um einen Spotmarkt. EPEX Spot wurde 2008 aus der französischen und deutschen Energiebörse (Powernext und EEX) gegründet. Beide Energiebörsen halten jeweils einen Anteil von 50% an EPEX Spot [13]. Der Stromverbrauch der Länder Deutschland, Frankreich, Österreich und Schweiz lag 2013 bei 1.200 TWh. Vergleicht man den Energieverbrauch mit dem der gesamten EU, so macht dieser Anteil nahezu 40% aus. Von diesen 1.200 TWh wurden rund 324 TWh durch EPEX Spot ausgehandelt [9]. Teilnehmer der EPEX Spot Börse sind beispielsweise Stadtwerke, Energieversorger wie EWE und E.ON und auch Banken, zum Beispiel die Deutsche Bank AG [1].

EPEX Spot lässt sich in zwei Märkten unterteilen; den Day-Ahead Markt und den Intraday Markt. Auf dem Day-Ahead Markt werden die Stromlieferungen für den nächsten Tag ausgehandelt. Der Handel findet unabhängig von Wochen- oder Feiertagen das ganze Jahr über statt. Dabei werden verschiedene Produkte angeboten, welche in Kapitel 3.3 genauer erläutert werden.

Auf dem Intraday Markt werden hingegen Stromlieferungen ausgehandelt, die noch am selben Tag geliefert werden. EPEX Spot ermöglicht die physische Lieferung bereits 45 Minuten nach abgeschlossener Auktion. Anders als beim Day-Ahead Markt kann hier die Stromlieferung eine Dauer von nur 15 Minuten betragen. So kann die Energie von Stromquellen, welche unregelmäßig viel Strom erzeugen, leichter eingespeist werden. Beispiele für solche Stromquellen sind Solar- oder Windkraftanlagen, bei denen die erzeugte Energie von Wetterbedingungen abhängt.

EPEX Spot versucht seine Leistungen qualitativ und quantitativ stetig zu verbessern. So wurde in der Pressemitteilung folgendes erwähnt: „Die heute gültigen 45 Minuten Vorlaufzeit könnten durch eine Feinjustierung der Intraday-Kette verkürzt werden. “ Und weiter heißt es: „Die Märkte müssen noch flexibler werden, andernfalls werden sich fluktuierende Energiequellen weiterhin den Marktmechanismen entziehen. Eine wirkungsvolle, vollständige Integration der Erneuerbaren ist notwendig für eine kosteneffiziente Energiewende.“ [2]. In dem Positionspapier aus dem Jahr 2014 ist protokolliert: „Auch der flexible Stromhandel an liquiden Intradaymärkten, u.a. mit Viertelstundenprodukten, ist ein effizienter Weg, fluktuierende Erneuerbare Energien kurzfristig in den Markt zu integrieren.“ [10]

Die Ziele, eine physische Stromlieferung nach abgeschlossener Auktion kurzfristiger zu gestalten und die Einbeziehung erneuerbarer Energien zeigen, dass EPEX Spot stets bemüht ist, aktuelle Themen aufzugreifen und den Handel voranzutreiben.

3.3 Produkte

EPEX Spot bietet verschiedene Produkte an, die jeweils ein bestimmtes Intervall innerhalb eines Tages definieren. Solch ein Intervall kann eine Länge von 15 Minuten bis 24 Stunden haben. Neben den Standartprodukten, welche in Abbildung 3.1 aufgelistet sind, gibt es für die Börsenteilnehmer die Möglichkeit, benutzerdefinierte Produkte auszuhandeln. Hierbei können die Intervalle beliebige Start- und Endzeitpunkte besitzen. Der Verkäufer verpflichtet sich, in diesem Intervall eine zuvor ausgehandelte Menge elektrischer Energie zu liefern. Die Mindestleistung liegt bei 100KW pro Produkt. Kleinere Leistungen werden nicht von EPEX Spot gehandelt.

Block **Baseload** für die Stunden 1 bis 24
Block **Peakload** für die Stunden 9 bis 20
Block **Night** für die Stunden 1 bis 6
Block **Morning** für die Stunden 7 bis 10
Block **High Noon** für die Stunden 11 bis 14
Block **Afternoon** für die Stunden 15 bis 18
Block **Evening** für die Stunden 19 bis 24
Block **Rush Hour** für die Stunden 17 bis 20
Block **Off-Peak 1** für die Stunden 1 bis 8
Block **Off-Peak 2** für die Stunden 21 bis 24
Block **Business** für die Stunden 9 bis 16
Block **Middle Night** für die Stunden 1 bis 4
Block **Early Morning** für die Stunden 5 bis 8
Block **Late Morning** für die Stunden 9 bis 12
Block **Early Afternoon** für die Stunden 13 bis 16
Block **Off-Peak** für die Stunden 1 bis 8, und 21 bis 24
Block **Sun-Peak** für die Stunden 11 bis 16

Abb. 3.1: Hier ist eine Liste aller Standartprodukte von EPEX Spot dargestellt. Zu jedem Produkt ist das entsprechende Intervall im Tagesverlauf angegeben. Quelle: [3].

In Abbildung 3.2 ist der Strombedarf über einen Tag verteilt schematisch dargestellt. Hierbei wird deutlich, dass die Stromnachfrage im Tagesverlauf nicht konstant ist, sondern Schwankungen aufzeigt. Die in der Abbildung zu sehende „langfristige Basisversorgung“ wird im Terminmarkt EEX ausgehandelt. Die „Baseload“ ist hingegen ein Standartprodukt von EPEX Spot und dauert von 0 bis 24 Uhr an. Anders als bei der langfristigen Basisversorgung endet die vertraglich geregelte Stromlieferung bei der Baseload um 24 Uhr. Am direkt darauf folgenden Tag gelten neue Baseload Produkte, welche am Vortag ausgehandelt wurden.

Wie sich an den Namen der Standartprodukte vermuten lässt, orientieren sich diese an den Bedarf zu bestimmten Tageszeiten. So beschreibt beispielsweise das Produkt „Peakload“ die Zeit von 9 bis 20 Uhr. In diesem Zeitraum ist der Strombedarf relativ hoch. Alle anderen Produkte aus Abbildung 3.1 lassen sich ebenfalls in Abbildung 3.2 eingliedern. Diese wurden jedoch aus Platzgründen nicht dargestellt.

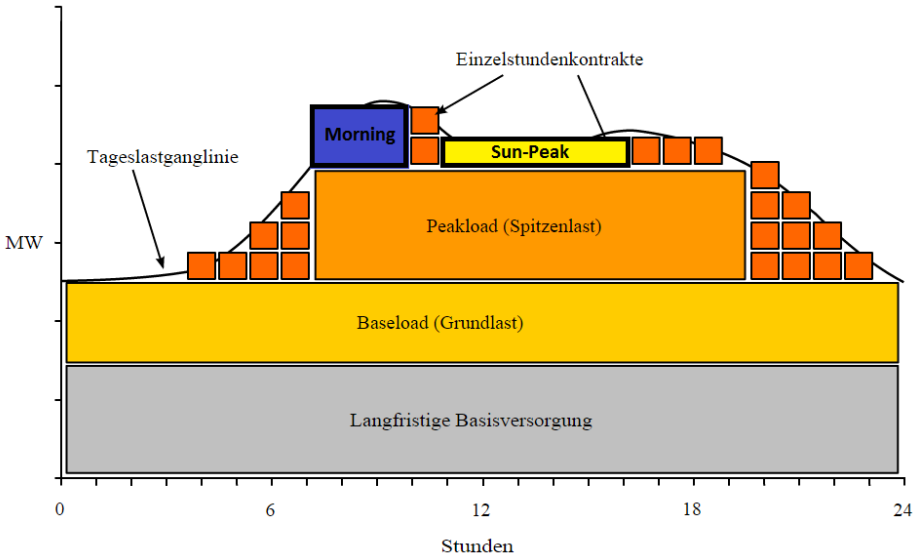


Abb. 3.2: Hier ist der Strombedarf über einen Tag verteilt schematisch dargestellt. Zusätzlich sind die Produkte „Baseload“, „Peakload“, „Morning“ und „Sun-Peak“ nach den Zeitintervallen eingegliedert. Quelle: [4] (Modifiziert).

Die Preise für die einzelnen Produkte ähneln sich zwar, sind jedoch nicht identisch. In Abbildung 3.3 sind die Preise der Produkte Baseload und Peakload im Zeitraum vom 01.06.2015 bis zum 07.06.2015 dargestellt.

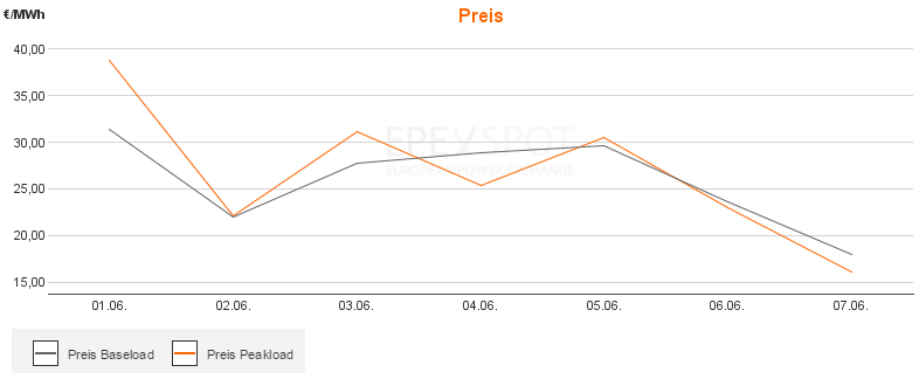


Abb. 3.3: Preise der Produkte Baseload und Peakload im Zeitraum vom 01.06.2015 bis zum 07.06.2015

Es lässt sich ein Trend für den Strompreis zu einem bestimmten Tag feststellen. Vergrößert man den betrachteten Zeitraum, so lässt sich feststellen, dass das Produkt Peakload tendenziell teurer ist, als das Produkt Baseload. In Abbildung 3.4 sind die Preise der Produkte Baseload und Peakload im Zeitraum vom 09.05.2015 bis zum 07.06.2015 in einer Grafik veranschaulicht. Die Preise dieser beiden Produkte können unter [5] bezogen werden.

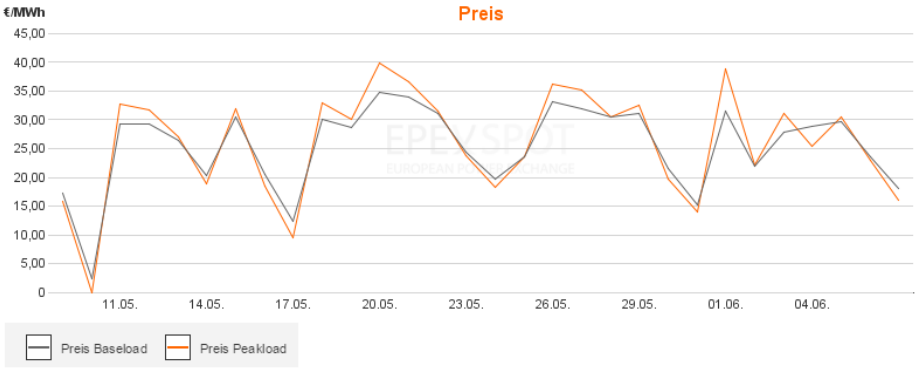


Abb. 3.4: Preise der Produkte Baseload und Peakload im Zeitraum vom 09.05.2015 bis zum 07.06.2015

3.4 Regelenergie

Im europäischen Stromnetz sind die Übertragungsnetzbetreiber bemüht, die Frequenz des Netzes stabil auf 50Hz zu halten. Wird mehr Strom in das Netz eingespeist als verbraucht, so steigt die Frequenz. Analog sinkt die Frequenz, falls mehr Strom verbraucht als eingespeist wird [15]. Um die Frequenz des europäischen Stromnetzes von 50Hz zu wahren, werden sogenannte Regelenergien eingesetzt. Diese Regelenergien gleichen Schwankungen im Stromnetz aus. Schwankungen entstehen, wenn unerwartet mehr, bzw weniger Energie verbraucht wird als erwartet. Allerdings können Schwankungen auch durch erneuerbare Energien entstehen, da hier die Menge der gewonnenen Energie nicht exakt geplant werden kann und von Wetterverhältnissen abhängt. So kann es beispielsweise passieren, dass erneuerbare Energien schlagartig mehr Energie in das Stromnetz einspeisen als zuvor prognostiziert. In solchen Fällen greift die Regelenergie ein.

Wird mehr Strom verbraucht als eingespeist, so liegt die Frequenz unter 50Hz. Es muss mehr Strom erzeugt werden, um auf die Soll-Frequenz von 50Hz zu gelangen. Wird zusätzlicher Strom zur Frequenzregulierung eingespeist, so spricht man von positiver Regelenergie. Wird jedoch mehr Strom produziert als verbraucht, so liegt die Frequenz über 50Hz und es muss mehr Strom verbraucht, bzw. weniger

Strom eingespeist werden, um auf die Soll-Frequenz von 50Hz zu gelangen. Wird elektrische Energie aus dem Netz genommen oder weniger Strom eingespeist, um die Frequenz zu stabilisieren, so ist von negativer Regelenergie die Rede.

Man unterscheidet 3 verschiedene Arten von Regelenergien: Primärreserve, Sekundärreserve und Minutenreserve [8]. Die Primärreserve greift ein, sobald der Toleranzwert von 0,01Hz über- oder unterschritten wird. Anbieter von Primärreserven verpflichten sich innerhalb von 30 Sekunden zu reagieren, sollte sich die Frequenz nicht im Toleranzbereich befinden. Die Bundesnetzagentur definiert die Primärreserve folgendermaßen: „Die Primärregelleistung dient der schnellen Ausregelung größerer Leistungsungleichgewichte im gesamten ENTSO-E-Verbund Kontinentaleuropa und wird solidarisch von allen beteiligten Regelzonen erbracht. “ [7] ENTSO-E steht für European Network of Transmission System Operators for Electricity und ist der Verbund der zentraleuropäischen Übertragungsnetzbetreiber. Abbildung 3.5 gibt an, zu welchem prozentualem Anteil die Primärreserve bei gemessener Spannung eingreifen muss.

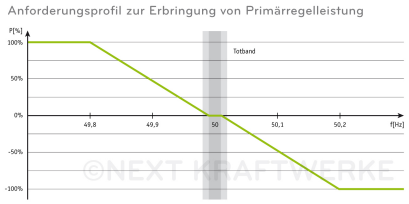


Abb. 3.5 Hier ist dargestellt, zu welchem prozentualem Anteil die Primärreserve bei gemessener Spannung eingreifen muss [6].

Die Sekundärreserve wird gleichzeitig mit der Primärreserve eingeleitet, jedoch muss die Sekundärreserve erst nach fünf Minuten 100% ihrer Leistung bereitstellen können. Ein Weiterer Unterschied ist, dass die Sekundärreserve von jedem einzelnen Übertragungsnetzbetreiber eingesetzt wird. Alle Übertragungsnetzbetreiber informieren sich gegenseitig über den Einsatz der Sekundärreserve, um alle Regelenergien gezielt in eine Richtung anzuwenden und sich gegenseitig ausgleichende Regulierungen zu vermeiden. Die Primär- und die Sekundärleistungen werden vom Regelenergieanbieter selbstständig angestoßen. Permanent wird die Netzfrequenz gemessen und falls notwendig auf Schwankungen reagiert.

Sollte die Netzfrequenz erheblich von dem Standardwert 50Hz abweichen, so greift die Minutenreserve unterstützend ein. Minutenreserven können innerhalb von 15 Minuten ihre volle Leistung erbringen.

Regelenergien werden auf Regelenergiemärkten ausgehandelt, nicht über EPEX Spot. Diese Regelenergiemärkte werden von Übertragungsnetzbetreibern betrieben. Um Regelenergie auf den Regelenergiemärkten anbieten zu können, müssen bestimmte Voraussetzungen erfüllt werden. Zunächst muss der Anbieter bei EPEX Spot als Stromlieferant registriert sein. Des weiteren ist der Anbieter dazu verpflichtet dem Übertragungsnetzbetreiber seine Einspeisungsprognosen für den Folgetag offen zu legen und eine gewisse Reservekapazität zur Verfügung zu stellen. Diese

Reservekapazitäten müssen sowohl für die positive, als auch für die negative Regelenergie verfügbar sein. Für die Primärreserve ist ein Mindestvolumen von 1 MW gefordert. Für die Sekundär- und Minutenreserve liegt die geforderte Reservekapazität bei 5 MW. [14]

Realisiert werden Regelenergien in der Regel durch Kraftwerke. Auch erneuerbare Energien können zur Regulierung eingesetzt werden: „Bereits heute bieten auf Erneuerbare Energien spezialisierte Stromhandelshäuser in einem geringen Umfang Regelleistung im Bereich der Sekundär- und Minutenreserve an, die durch Biogasanlagen bereitgestellt wird.“ [11] Diese können durch das Hoch- und Herunterfahren der Anlagen gezielt und schnell die Stromerzeugung regulieren. Kraftwerke, die Regelenergien anbieten, erhalten allein durch die Bereitstellung eine Bereitschaftsgebühr. Kommt es nun zum Bedarf an Regelenergie, so erhält das Kraftwerk zusätzlich für die Regulierung eine Leistungsgebühr.

Literaturverzeichnis

[1] Zuletzt Besucht: [29.05.2015]. URL: http://www.epexspot.com/en/membership/list_of_members/participants/trading.

[2] Zuletzt Besucht: [30.05.2015]. URL: http://www.epexspot.com/de/presse/press-archive/details/press/Flexible_M_rkte_sind_Schl_ssel_zu_effizienter_Energiewende.

[3] Zuletzt Besucht: [29.05.2015]. URL: https://www.epexspot.com/en/product-info/auction.

[4] Zuletzt Besucht: [29.05.2015]. URL: http://commons.wikimedia.org/wiki/File:Stromb%C3%83%C2%B6rse_Stromverbrauch_Lastprofil.png.

[5] Zuletzt Besucht: [07.06.2015]. URL: https://www.epexspot.com/de/marktdaten/dayaheadauktion/chart/auction-chart/.

[6] Zuletzt Besucht: [30.05.2015]. URL: https://www.next-kraftwerke.de/wissen/regelenergie/primaerreserve.

[7] Bundesnetzagentur. „Beschluss Az: BK6-10-098“. In: (2011).

[8] Frontier Economics. „Strommarkt in Deutschland - Gewaehrleistet das derzeitige Marktdesign Versorgungssicherheit?“ In: (2014).

[9] EPEX-SPOT. „EPEX SPOT Facts and Figures“. In: (2013).

[10] EPEX-SPOT. „Positionspapier der European Energy Exchange und EPEX SPOT“. In: (2014).

[11] Philipp Schaube Hans-Jochen Luhmann Dorothea Schostok. *Vom Inter- zum Intra-Wettbewerb - Stufen der Integration Erneuerbarer Energien im Strombereich*. Springer, 2014.

[12] Ralf Schaefer Ines Zenke. *Energiehandel in Europa*. C.H. Beck, 2009.

[13] Panos Konstantin. *Praxisbuch Energiewirtschaft*. Springer Vieweg, 2013.

[14] Jan Paulus. „Was ist und wie funktioniert eigentlich die Praequalifikation fuer den Regelenergiemarkt“. In: (2013).

[15] Christoph Pieper u. a. „Die wirtschaftliche Nutzung von Power-to-Heat-Anlagen im Regelenergiemarkt“. In: (2015).

Kapitel 4
Epex Spot: Preisbildung

Jan Korte

Zusammenfassung Der Strommarkt war bisher in viele kleine Anbieter aufgeteilt. Mit der Strombörse Epex Spot werden diese länderübergreifend gekoppelt. Doch wie kommt der Strompreis zustande? Mithilfe eines komplexen Algorithmus wird auf die Bedingungen der verschiedenen Stromprodukte eingegangen. In der folgenden Arbeit werden zunächst die Grundlagen von Preisbildung an Börsen geschaffen. Dann wird die Merit-Order vorgestellt, ein vereinfachtes Verfahren zur Berechnung des Strompreises. Und letztlich wird der Preisbildungs-Algorithmus von Epex Spot detailliert erläutert.

4.1 Einleitung

Als erstes wird die Preisbildung an Börsen erläutert. Dann wird die Merit-Order, ein vereinfachtes Verfahren zur Berechnung des Strompreises vorgestellt. Auf der Epex Spot gibt es verschiedene Auftragstypen bzw. Stromprodukte. Diese werden erläutert, bevor im letzten Kapitel der Preisbildungs-Algorithmus selbst vorgestellt wird. Die Auftragstypen fügen dem Preisbildungs-Algorithmus schwierig lösbare Bedingungen hinzu. Deshalb wird das Hauptproblem der Preisfindung in Sub-Probleme unterteilt.

4.2 Preisbildung an Börsen

Der Preis einer bestimmten Aktie an der Börse hängt von Angebot und Nachfrage ab. Im Folgenden wird mithilfe eines Beispiels erläutert, wie der Handel und damit auch der Preis zustande kommt.

Carl von Ossietzky Universität Oldenburg
E-mail: jan.korte@uni-oldenburg.de

Der Börsenmakler nimmt alle Kauf- und Verkaufswünsche (englisch *orders*) entgegen. Er errechnet, bei welchem Kurs die meisten Wünsche erfüllt werden. Dies ist dann das optimale Verhältnis und somit der Börsenkurs.

Ein Börsenpreis von **98 €** würde unterhalb aller Kaufwünsche, aber nur oberhalb von einem Verkaufswunsch liegen. Es würden also nur die 10 Aktien dieses Verkaufswunsch verkauft werden:

Kaufen		Verkaufen	
Menge	Preis	Menge	Preis
10	99,50 €	10	98 €
20	100 €	30	99,50 €
20	101 €	10	100,50 €

Ein Börsenpreis von **101 €** würde überhalb aller Verkaufswünsche, aber nur unterhalb von einem Kaufwunsch liegen. Es würden also nur die 20 Aktien dieses Kaufwunsches gehandelt werden:

Kaufen		Verkaufen	
Menge	Preis	Menge	Preis
10	99,50 €	10	98 €
20	100 €	30	99,50 €
20	101 €	10	100,50 €

Ein Börsenpreis von **100 €** hingegen ist das optimale Verhältnis aus Käufen und Verkäufen. Hier würden 40 Aktien verkauft werden:

Kaufen		Verkaufen	
Menge	Preis	Menge	Preis
10	99,50 €	10	98 €
20	100 €	30	99,50 €
20	101 €	10	100,50 €

4.3 Strompreisbildung

In diesem Kapitel werden zwei Konzepte zur Bildung des Strompreises erläutert. Das erste Konzept ist die Merit-Order, welche eine eher abstrakte und vereinfachte Sicht auf den Strommarkt ist. Das zweite Konzept, die Strombörse EPEX, entspricht im Prinzip der Merit-Order, hat aber eine präzisere Sicht auf die Stromangebote.

4.3.1 Merit-Order

Die Merit-Order (*merit* = Leistung) ist die Einsatzreihenfolge der Kraftwerke. In Abb. 4.1 wird diese dargestellt. Dazu werden alle Kraftwerks anhand ihrer Grenzkosten sortiert. Die Grenzkosten sind die Kosten, die durch die Produktion einer zusätzlichen Mengeneinheit eines Produktes entstehen.

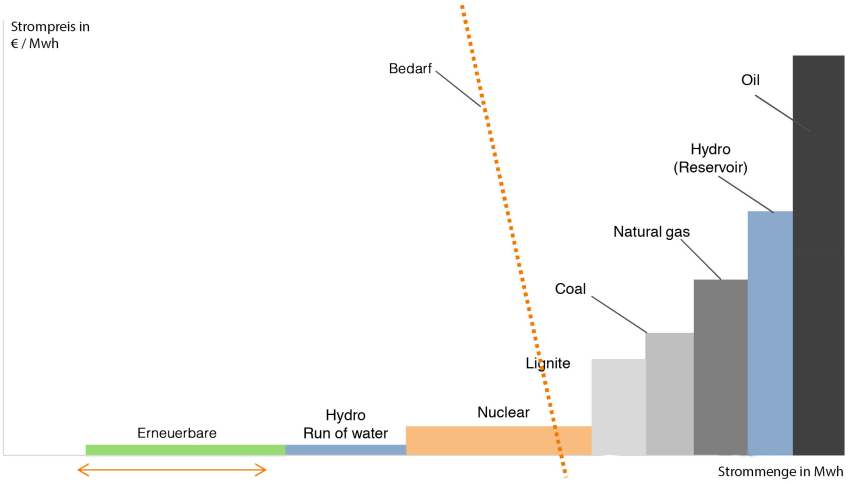


Abb. 4.1: Merit-Order

Dabei ist die Breite einer Säule ist die verfügbare Strommenge der jeweiligen Kraftwerksart. Außerdem gibt es eine Nachfrage-Kurve. Sie verläuft schräg, weil ein geringer Strompreis zu höherer Nachfrage führt. Der Schnittpunkt der Nachfrage-Kurve mit dem Strompreis der Kraftwerke ist der Börsenpreis. Alle Verkäufe finden zu diesem Börsenpreis statt.

Der Börsenpreis wird beeinflusst durch die Vorhersage des Bedarfs sowie durch die beteiligten Kraftwerke bzw. den an die Kraftwerke gekoppelten Rohstoffpreisen.

Die Erneuerbaren sind ganz unten, weil sie einerseits sehr niedrige Grenzkosten haben und andererseits eine erhöhte Priorität bekommen. Denn bei Gaskraftwerken kann z. B. vergleichsweise leicht so viel produziert werden, wie gebraucht wird, aber der Strom der erneuerbaren Energien wird in jedem Falle produziert und muss verkauft werden.

Weil die tatsächliche Produktion der erneuerbare Energien nicht präzise vorausgesagt werden kann, verändert sich der Börsenpreis z. B. bei wenig Sonne. Weil alle

erneuerbaren Energien verkauft werden, verursachen sie so eine gewisse Fluktuation auf dem Strommarkt.

Insgesamt kann ein hoher Anteil von erneuerbaren Energien den Börsenpreis senken, weil durch den hohen Anteil die teureren Kraftwerke verdrängt werden.

4.3.2 EPEX Strompreisbildung

Die Inhalte der folgenden Kapitel wurden mit Anlehnung an [1] verfasst.

Das Price Coupling of Regions (PCR) ist ein Projekt sieben großer Strombörsen europäischer Länder. Sie entwickelten einen einzigen, gemeinsamen Preis-Kopplungs-Algorithmus namens EUPHEMIA (Pan-European Hybrid Electricity Market Integration Algorithm).

Der Hauptvorteil des länderübergreifenden Koppelns einzelner, kleiner Märkte ist eine bessere Marktliquidität. Das bedeutet, dass Strom jederzeit gehandelt werden kann, ohne dass dadurch der Marktpreis sinkt. Als Nebeneffekt sind die Strompreise weniger sprunghaft.

Das nun erklärte Konzept EUPHEMIA's ist die Grundlage für die folgenden Kapitel.

Als erstes werden Angebot und Nachfrage pro Zeitperiode und *bidding area* aggregiert (Abb. 4.2). Eine *bidding area* ist der geographisch jeweils kleinste lokale Markt für *orders*. Dies könnte beispielsweise der Landkreis Oldenburg sein. Als zweites wird der Börsenpreis pro *bidding area* und Zeitperiode berechnet.

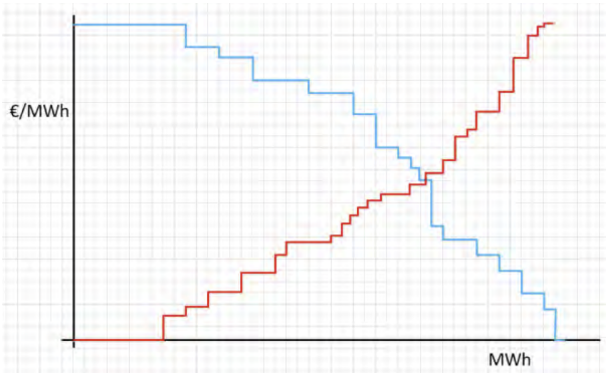


Abb. 4.2: Nachfrage nach Preis absteigend, Angebote aufsteigend sortiert

Es werden nun alle *orders* akzeptiert, für die gilt:

- Der Nachfrage-Preis größer als der Börsenpreis, oder
- der Angebots-Preis ist kleiner als der Börsenpreis.

Falls Anpassungen des Börsenpreises möglich sind (siehe Abb. 4.3), wird dieser so gelegt, dass er für Käufer und Verkäufer gleich fair ist und dass möglichst viele Käufe bzw. Verkäufe stattfinden.

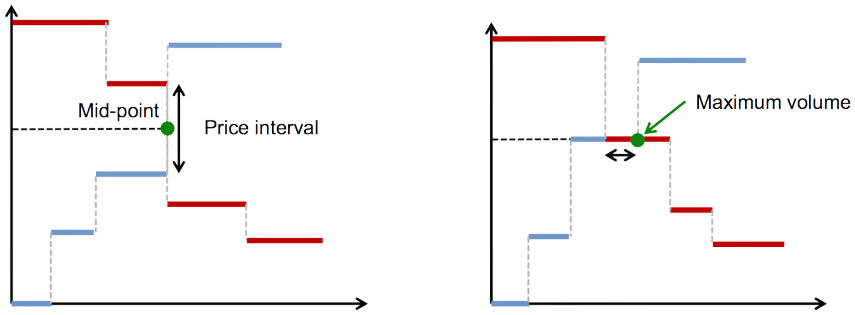


Abb. 4.3: Anpassungen bei der Berechnung des Börsenpreises

Das Ergebnis eines Tages ist in Abb. 4.4 zu sehen.

4.4 Auftragstypen

Um auf die Anforderungen der Kraftwerksbesitzer eingehen zu können, gibt es verschiedene Typen von Aufträgen. Die Wichtigsten davon werden in den folgenden Kapiteln näher erläutert:

- *block order*
- *merit order*
- *Minimum Income Condition order*
- *Prezzo Unico Nazionale*

4.4.1 block order

Eine *block order* kann entweder auf der Angebotsseite oder auf der Nachfrageseite sein. Sie wird definiert durch

- einen Minimalpreis für Angebote oder einen Maximalpreis für Gebote,
- eine beliebige Anzahl Zeiträume (des nächsten Tages),

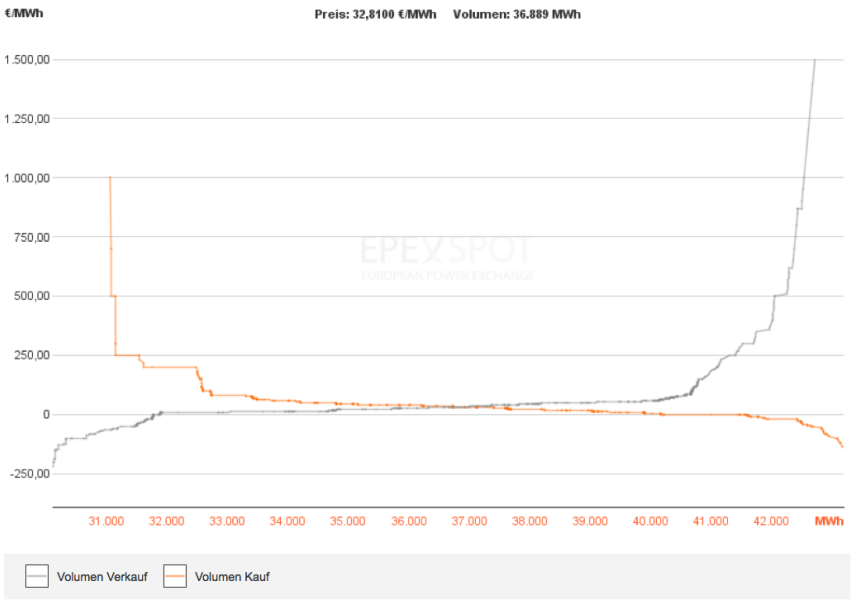


Abb. 4.4: Aggregation von Angebot und Nachfrage vom 04.05.2015

- eine bestimmten Strommenge pro Zeitraum und
- eine minimalen Akzeptanz-Rate.

Die Akzeptanz-Rate ist in der Regel 1, es muss dann der gesamte Strom verkauft werden. Bei einer Akzeptanz-Rate von 0,5 müsste nur die Hälfte verkauft werden. Falls das Angebot über mehrere Zeiträumen aufgeteilt ist, wird es akzeptiert, wenn der Durchschnittspreis erreicht wurde.

In Abb. 4.5 sind zwei exemplarische *block orders* zu sehen.

Type	PERIOD	PRICE	VOLUME
BLOCK BUY	Hours 1-24	40 Euros	200 MWh
BLOCK SELL	Hours 8-12	40 Euros	50 MWh

Abb. 4.5: Jede Zeile ist ein Beispiel einer *block order*.

Bei Bedarf kann eine *block order* auf folgende Arten verfeinert werden:

- Das Angebot wird nur akzeptiert, wenn ein bestimmtes anderes Angebot auch akzeptiert wurde.

- Mehrere Angebote können in einer exklusiven Gruppe sein. Wird ein Angebot daraus verkauft, sind die anderen nicht mehr verfügbar.

4.4.2 merit order

Eine *merit order* ist eine *block order*, welcher eine Nummer zugeteilt wird. *Merit orders* werden genutzt, um bei ähnlichen Angeboten die mit der höheren Priorität zu bevorzugen. Das können z. B. Windkraftwerke sein. Je geringer die Nummer, desto höher ist die Priorität. Wenn mehrere Angebote innerhalb des Börsenpreises sind und in benachbarten, nicht überfüllten *bidding areas* liegen, dann wird das Angebot mit der geringeren Nummer akzeptiert. Mithilfe der *merit order* könnte der Staat Kraftwerke mit erneuerbaren Energien oder Kraftwerke kleinerer Anbieter bevorzugen.

4.4.3 Minimum Income Condition order

Die *Minimum Income Condition order* ist nur für Angebote möglich. Sie hat als Bedingung, dass der erzielte Preis die Produktionskosten decken muss. Der finale Preis ist die Summe aus

- den fixen Startkosten des Kraftwerks und
- den Produktionskosten (Preis pro bestellte Mwh).

Eine *Minimum Income Condition order* wird ganz oder gar nicht akzeptiert.

4.4.4 Prezzo Unico Nazionale

In Italien verursachen unterschiedlich weit entwickelte Regionen ein Preisgefälle innerhalb des Landes. Die beiden Hauptgründe dafür sind Stromerzeugungs-Monopole und zu geringer Netzausbau. Um die Situation für den Verbraucher angenehmer zu machen, gibt es den *Prezzo Unico Nazionale* (PUN). Das ist der Einheitspreis auf dem *Day-ahead market*. Alle italienischen *orders* haben also zusätzlich den Typ *PUN order*. Ein PUN-Angebot wird verkauft, wenn der Verkaufspreis über dem aktuellen PUN liegt. Dabei kann der Verkaufspreis sogar unter dem Börsenpreis liegen. Der Produzent bekommt bei Verkäufen den Preis der jeweiligen *bidding area*.

Dazu ein Beispiel in einer *bidding area* in Italien:

- Angebot: 4 €
- Nachfrage: 6 €
- Börsenpreis: 5 €

- PUN: 8 €

→ Darf nicht für 5 € verkauft werden, da unterhalb des PUN.

4.5 Algorithmus

Das Ziel des Preisbildungs-Algorithmus ist die Berechnung folgender Daten für jede Zeitperiode des Tages:

- Börsenpreis pro *bidding area*
- *net position* (Angebot minus Nachfrage) pro *bidding area*
- Strom-Durchflüsse pro Verbindung
- verkaufte Strommenge

Der Algorithmus wird täglich von der Epex Spot ausgeführt, nachdem alle Angebote und Gebote eingegangen sind. Er berechnet den Handel des nächsten Tages. Mit der Veröffentlichung der Ergebnisse gehen die Anbieter der angenommen Angebote einen Vertrag mit ihren Käufern ein. Es wird sozusagen ein „Fahrplan“ für den nächsten Tag aufgestellt. In Abb. 4.6 ist das Ergebnis eines Tages zu sehen.

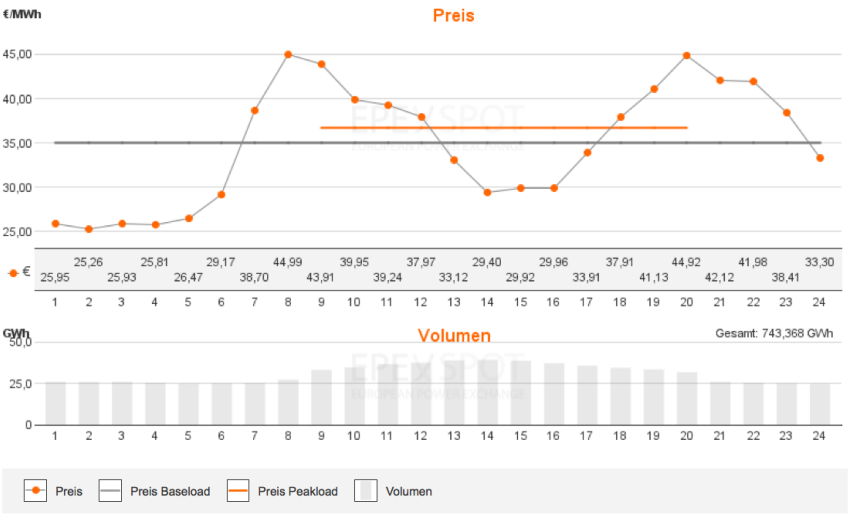


Abb. 4.6: Ergebnis des Algorithmus vom 04.05.2015. Oben: Strompreis pro Stunde am Tag. Unten: Verkaufte Strommenge pro Stunde am Tag

4.5.1 Constraints

Das Handelsvolumen (*welfare*) soll unter Berücksichtigung der folgenden Punkte maximal werden:

- *merit order*: *Orders* mit niedriger Nummer werden bevorzugt.
- *PUN order*: In Italien sind alle akzeptierten *orders* oberhalb des PUN.
- *block order*: Die Nebenbedinungen wie z. B. die Akzeptanzrate sind erfüllt.
- *Minimum Income Condition orders* werden nur angenommen, wenn sie den erwünschten Preis erzielen.
- Der stattfindende Stromfluss ist innerhalb der Netzwerk-Kapazität.

4.5.2 Probleme

Das zu lösende Problem wird vereinfacht, indem zunächst alle Constraints außer die von *block order* und *Minimum Income Condition order* ignoriert werden. Diese beiden verbleibenden *order* Typen sind vergleichsweise einfach modellierbar: Sie werden entweder angenommen oder abgelehnt. Diese Eigenschaft wird mit einem Boolean dargestellt.

Das gerade beschriebene Problem wird *Welfare Maximization Problem (Master Problem)* bezeichnet. Sobald es eine Lösung findet (also für jede *order* entscheidet, ob sie akzeptiert oder abgelehnt wird), wird das nächste Constraint genommen und die vorhandene Lösung bezüglich dieses Constraints überprüft. Eine Übersicht dieses Prozesses ist in Abb. 4.7 zu sehen.

Neben dem Master-Problem gibt es noch drei weitere Probleme:

Das Price Determination Sub-Problem berechnet den Börsenpreis pro *bidding area*. Dabei wird überprüft, ob die besonderen Bedingungen der *block* und *MIC orders* durchführbar sind. Zusätzlich wird noch überprüft, ob die Stromfluss innerhalb der Netzwerk-Kapazität ist.

Beim PUN Search Sub-Problem wird pro Stunde des Tages ein PUN Preis gefunden, der möglichst viele *PUN orders* akzeptiert. Möglicherweise wurden nun durch die *PUN orders* einige komplexere *block orders* verletzt. Deshalb wird die Richtigkeit der *block orders* erneut überprüft.

Für die bereits gefundene Lösung – bestehend aus Börsenpreisen und akzeptierten *orders* – kann es unterschiedliche Konstellationen von *net positions* (Angebot minus Nachfrage pro *bidding area*) geben. Außerdem kann es mehrere mögliche Wege geben, auf denen der Strom von Erzeuger zu Verbraucher fließt. Das Volume Indeterminacy Sub-Problem berechnet zur vorhandenen Lösung die *net positions*

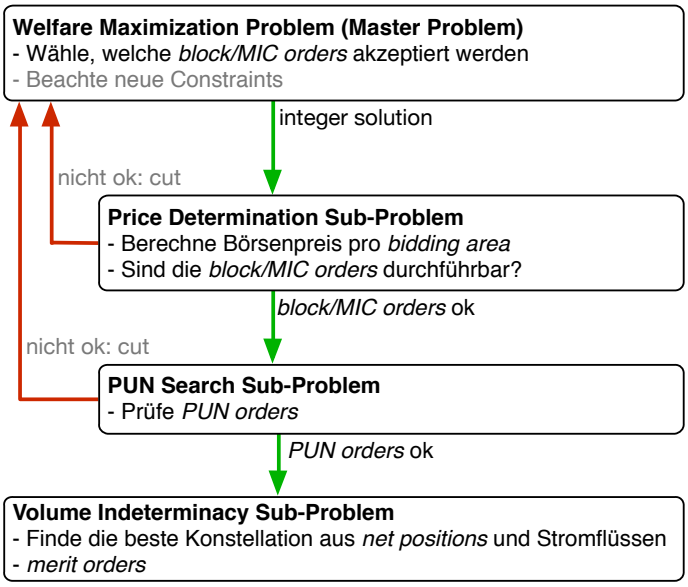


Abb. 4.7: Ablauf und Teilprobleme des Algorithmus

und Stromflüsse mit höchstem Handelsvolumen. Bei dieser Berechnung werden nun auch die *merit orders* einbezogen, also Angebote mit niedriger *merit order*-Nummer bevorzugt.

4.5.3 Ablauf

Die Probleme werden mithilfe eines kombinatorischen Optimierungsprozesses gelöst. Dabei soll aus einer Menge von ganzzahligen Elementen (alle *orders*) eine Teilmenge gefunden werden (die *orders*, die angenommen werden), welche bestimmte Constraints erfüllt und bezüglich einer Kostenfunktion optimal ist (maximaler *welfare*). Ganzzahlig bedeutet, dass eine gewöhnliche *block order* entweder ganz oder gar nicht angenommen wird. Es ist leicht, eine Lösung mit maximalem Handelsvolumen zu finden, wenn Blöcke auch zum Teil angenommen werden dürfen. Ein Verfahren, welches ganzzahlige Lösungen findet, nennt sich Branch-and-Cut.

Ein Branch-and-Cut Algorithmus läuft im Allgemeinen wie folgt ab: Als erstes wird versucht, das Problem relaxiert zu lösen, z. B. in dem auch gebrochene Lösungen erlaubt sind. Sobald eine Variable aus der Lösungs-Teilmenge jedoch nicht mehr ganzzahlig ist, gibt es zwei Möglichkeiten:

- Der aktuelle Lösungsversuch wird verzweigt indem das Problem in zwei Teilprobleme unterteilt wird (*Branch*). Die Variable die eigentlich ganzzahlig sein sollte, wird dann in zwei Bereiche aufgeteilt. Für jeden Bereich wird dann mit der Beschränkung auf dieser Variable erneut nach Lösungen gesucht.
- Der aktuelle Lösungsraum wird durch einen *Cut* weiter eingeschränkt, indem eine bestimmte Menge von möglichen Lösungen aus dem Lösungsraum ausgeschlossen wird. Mit dieser Beschränkung wird erneut nach Lösungen gesucht.

Mit jeder Beschränkung des Problems können andere, der Kostenfunktion näher liegende Lösungen gefunden werden.

Im Folgenden wird erläutert, wie Branch-and-Cut auf die Probleme des Preisbildungs-Algorithmus angewandt werden kann.

Zunächst wird eine mögliche Lösung für das Master-Problem (akzeptierte / abgelehnte *orders*) gesucht. In den meisten Fällen wird keine ganzzahlige Lösung gefunden, es gibt also gebrochene Blöcke. Von diesen gebrochenen Blöcken wird einer ausgewählt und ein Branch erstellt: Im ersten Zweig wird der gebrochene Block komplett abgelehnt und dann der *welfare* berechnet. Im zweiten Zweig wird der Block akzeptiert und dann der *welfare* berechnet. Bei der Berechnung des *welfare* wird die Strommenge aller angenommenen *orders* addiert.

Es kann jedoch sein, dass durch den Branch weitere gebrochene Blöcke entstehen. Dann wird ein Cut ausgeführt, d. h. dem Master-Problem ein weiteres Constraint hinzugefügt und erneut nach Lösungen gesucht. Diese Lösungen sind nun näher am optimalen Ergebnis.

Zudem wird bei jedem Lösungsversuch der *welfare* gespeichert. Da durch Constraints die optimale, aber gebrochene Lösung eingeschränkt wird, ist der berechnete *welfare* eine Obergrenze für nachfolgende Lösungsversuche. Auf diese Weise muss ein Zweig nicht weiter ausgeführt werden, wenn der *welfare* des Zweigs unterhalb des *welfare* einer funktionierenden Lösung ist.

Wenn eine bestimmte Konstellation akzeptierter Blöcke keine Lösung bringt, wird diese Konstellation als Constraint dem Master-Problem hinzugefügt (Cut). Cuts können aber auch später innerhalb des Lösungsfindungsprozesses auftreten, beispielsweise wenn einige *PUN orders* fälschlicherweise angenommen werden. Ein beispielhafter Ablauf des Branch-and-Cut Verfahrens ist in Abb. 4.8 zu sehen.

Da der Algorithmus in der Produktion genutzt wird, bricht er ab, sobald alle Knoten erforscht wurden oder das Zeitlimit erreicht wurde. Er nutzt das Zeitlimit jedoch aus, um nach dem Finden einer Lösung eine besser Lösung zu finden.

Beim Branch-and-Cut Algorithmus ist es nicht garantiert, dass innerhalb einer gegebenen Zeitschranke ausreichend gute Lösungen gefunden werden. Meistens wird das jedoch der Fall sein.

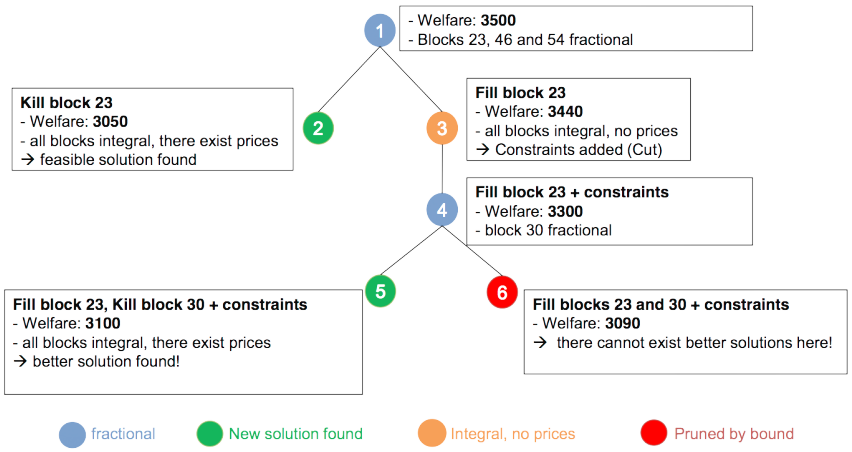


Abb. 4.8: Vereinfachter Ablauf des Branch-and-Cut Verfahrens

Das Preisfindungs-Problem ist vermutlich NP-vollständig. Innerhalb des mitgegebenen Zeitrahmens (z. B. 10 Minuten) wird aber trotzdem eine ausreichend gute Lösung gefunden.

4.6 Zusammenfassung

Zunächst wurde gezeigt, wie der Börsenkurs als optimales Verhältnis der Kaufs- und Verkaufswünsche gebildet wurde. Dann wurde ein vereinfachtes Verfahren zur Berechnung des Strompreises vorgestellt: Bei der Merit-Order wird das Stromangebot nach Preis bzw. nach Stromart sortiert. Der Schnittpunkt mit der Nachfrage-Kurve ist der Börsenpreis.

Zur Erklärung des Algorithmus zur Preisbildung auf der Epex Spot wurden die verschieden *order*-Typen mit ihren Bedingungen vorgestellt. Da das Problem zu komplex ist, wurde es in Sub-Probleme aufgeteilt, die einfacher zu lösen sind. Der Algorithmus selbst nutzt Branch-and-Cut um den Lösungsraum nach und nach einzuschränken.

Durch die länderübergreifende Kopplung kleinerer Strommärkte kann der Strom nun einfacher gehandelt werden, der Strompreis ist weniger sprunghaft. Zudem bietet eine Verlagerung des Stromhandels weg von langfristigen Verträgen hin zum *Day-ahead market* einen besseren Absatz für neue, kleinere Kraftwerke.

Literaturverzeichnis

[1] EPEX Spot – APX – Belpex – Nord Pool Spot – OMIE - Mercatoelettrico (GME) – OTE. „EUPHEMIA Public Description - PCR Market Coupling Algorithm“. In: (2013). letzter Zugriff 01.07.2015. URL: http://www.epexspot.com/de/Marktkopplung/PCR_Price_Coupling_of_Regions.

Kapitel 5
Portfoliooptimierung

Lars Elend

Zusammenfassung Die *Portfoliooptimierung* ist eine insbesondere von Markowitz „Portfolio Selection“ geprägte Optimierung zur Auswahl von Finanzinstrumenten, wie z. B. Wertpapieren. Die Portfoliooptimierung kann jedoch auch auf andere Bereiche, wie den Elektrizitätsmarkt angewendet werden. Die Ziele der Optimierung sind die Maximierung des erwarteten Ertrags und die Minimierung des damit verbundenen Risikos. Nach der Beschreibung der Portfoliooptimierung wird auf die Kritik von Taleb zur Normalverteilungsannahme eingegangen, die zur Abschätzung des Risikos genutzt wird.

5.1 Einleitung

Die Portfoliotheorie nach Markowitz beschäftigt sich mit der Auswahl eines für einen bestimmten Anleger optimalen Portfolios auf dem Finanzmarkt. Jenes kann mit Hilfe der *Portfoliooptimierung* gefunden werden. Dazu werden als Ziele zum einen die Maximierung des erwarteten Ertrages und zum anderen die Minimierung des damit verbundenen Risikos berücksichtigt.

Dieser Zusammenhang wird im Grundlagenabschnitt weiter erläutert. Danach wird die Portfoliooptimierung in Abschnitt 5.3 formal definiert und zusätzlich mit einer graphischen Darstellung visualisiert. Schließlich wird der Bezug zum Elektrizitätsmarkt in Abschnitt 5.4 hergestellt. In Abschnitt 5.5 wird die Kritik von Taleb an der Normalverteilungsannahme erörtert, die ein wesentlicher Bestandteil der Portfoliooptimierung ist. Er weist auf das Auftreten von *Schwarzen Schwänen* hin, aber gibt mit der *Antifragilität* auch einen Lösungsansatz. Zuletzt wird in Abschnitt 5.6 eine Zusammenfassung gegeben.

Carl von Ossietzky Universität Oldenburg
E-mail: lars.elend@uni-oldenburg.de

5.2 Grundlagen

Ein *Portfolio* beschreibt eine Zusammenfassung von Objekten eines bestimmten Typs. So steht es auf dem Finanzmarkt für eine Ansammlung von Wertpapieren, die in diesem Verbund gemeinsam gehandelt werden können. Die *Portfoliotheorie* (engl. Modern portfolio theory (MPT)) hat zum Ziel, den erwarteten Gewinn eines Portfolios zu maximieren und das Risiko zu minimieren. Zusätzlich wird auch die Korrelation zwischen den Wertpapieren berücksichtigt und als Kovarianz ausgedrückt.

Markowitz stellte sich die Frage, wie ein Investor einen Punkt auf der, aus den beiden Zielen entstehenden, Pareto Front, die nun auch als *effiziente Front* bezeichnet wird, auswählen würde [3, S. 470]. Die Pareto Front ist die Menge der Punkte im Zielraum, die nicht dominiert werden. Ein Punkt wird dominiert, falls es einen anderen gibt, der in mindestens einem Ziel besser, aber in keinem schlechter ist. Die effiziente Front wird in Abb. 5.1 visualisiert [2]. Dafür wurden die beiden Ziele auf die Achsen aufgetragen: Erwarteter Ertrag ($E(r)$) und Risiko (σ). In grau ist die Menge aller Portfolios dargestellt. Die am weitesten links liegenden, bilden die *Minimum Varianz Front* (rot und grün). Diese entspricht einem Anleger, der versucht, ein möglichst geringes Risiko einzugehen. Da auf dieser Front jedoch jeweils zwei Punkte mit dem gleichen Risiko existieren, wird nun auch der erwartete Ertrag berücksichtigt: Ein höherer Ertrag wird bevorzugt. So entsteht die *effiziente Front*. Diese kann schließlich mit der *Portfoliooptimierung* berechnet werden, die im nächsten Abschnitt beschrieben wird.

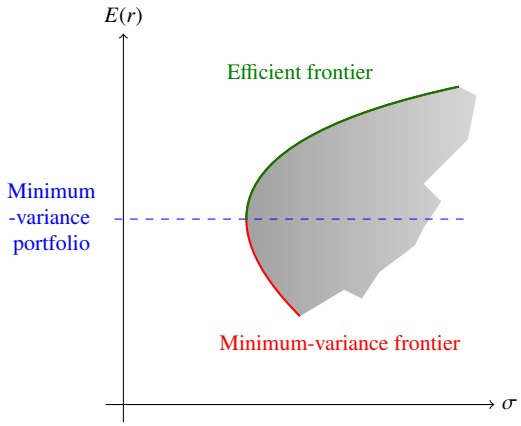


Abb. 5.1: Effiziente Front

Durch die Diversifizierung auf verschiedene Wertpapiere wird das Risiko minimiert. Dies entspricht auch Bernoulli's *Gesetz der großen Zahlen* (1713) [6]: Durch

die Erhöhung der Anzahl der Wertpapiere hat jedes einzelne einen geringeren Einfluss auf das Gesamtrisiko und es wird sich einem Mittelwert angenähert. Außerdem wird die Aussage „Das Ganze ist mehr als die Summe seiner Teile“ hiermit erfüllt, da nur der Zusammenschluss von vielen Wertpapieren das Risiko verringern kann.

In der Portfoliotheorie werden einige Annahmen getroffen, die großenteils mit dem theoretischen Modell des *vollkommenen Kapitalmarktes* zusammengefasst werden. Zum einen wird angenommen, dass alle Investoren rational handeln und versuchen ihren Gewinn anhand von allgemein bekannten Informationen, wie Erwartungswert, Varianz und Kovarianz, zu maximieren und das Risiko zu meiden. Enthalten darin ist auch die Normalverteilungsannahme über die erwarteten Erträge, welche im Abschnitt 5.5 kritisiert wird. Zum anderen haben die Investoren keinen Einfluss auf Preisänderungen; die Preise bleiben konstant. Auch die Korrelationen zwischen verschiedenen Wertpapieren werden als konstant angenommen. Des Weiteren werden Transaktionskosten, wie Steuern oder Informationskosten, nicht berücksichtigt. [1]

Da durch die zahlreichen Annahmen das Ergebnis verfälscht werden kann, wurden seitdem noch einige Erweiterungen zu der Portfoliotheorie entwickelt. So gibt es beispielsweise die *Post-modern portfolio theory (PMPT)*, bei der keine Normalverteilung genutzt wird um ein asymmetrisches Risiko abbilden zu können. Eine andere Weiterentwicklung ist das *Black-Litterman-Verfahren*, welches 1990 bis 1992 von Fischer Black und Robert Litterman entwickelt wurde. Dabei handelt es sich um ein mathematisches Modell zur Berechnung der Anlageaufteilung (Zusammensetzung des Portfolios). Zwar kann auch die Portfoliotheorie nach Markowitz dieses Problem lösen, jedoch ist das Abschätzen des erwarteten Ertrages in der Praxis schwierig. Daher wird beim Black-Litterman-Verfahren auf diesen Eingabeparameter verzichtet, indem zunächst mit dem Kapitalgutpreismodell (engl. Capital Asset Pricing Model (CAPM)) ein Renditevektor berechnet wird.

5.3 Portfoliooptimierung

Im Folgenden wird die Portfoliooptimierung formal definiert. Die Definition orientiert sich großenteils an dem Paper von Liu und Wu [2].

Zunächst werden die vom Portfolio beinhalteten Vermögenswerte (assets) in risikobehaftete und risikofreie eingeteilt. Ohne Beschränkung der Allgemeinheit können alle risikofreien Vermögenswerte zu einem einzigen akkumuliert werden. So bekommen die risikobehafteten assets die Indizes 1 bis n und das risikofreie den Index $n+1$. Die Ertragsrate (rate of return) r_i der einzelnen Portfolios ($1 \leq i \leq n+1$) kann durch ein mit w_i gewichtetes Mittel den Ertrag des Portfolios bilden:

$$r_p = \sum_{i=1}^{n+1} w_i r_i \tag{5.1}$$

Da die Bestandteile des Portfolios risikoreich sind, ist ihr Ertrag und somit auch der Ertrag des Portfolios nicht sicher. Daher wird Wissen über die zugehörige Wahrscheinlichkeitsverteilung benötigt. Dann bezeichnet der Erwartungswert den erwarten Ertrag ($E(r_p)$) und die Varianz (σ^2), die die Streuung der Verteilung beschreibt, wird als Indikator für das Risiko genutzt. Dies bildet die Grundlage des von Markowitz und Tobin entwickelten *mean-variance criterion (MVC)*.

Die obige Formel (5.1) wird also um eine Erwartungswertfunktion ergänzt. Für die Varianz wird auch die Korrelation zwischen zwei Vermögenswerten (i und j) mit Hilfe der Kovarianz σ_{ij} berücksichtigt.

$$E(r_p) = \sum_{i=1}^{n+1} w_i E(r_i) \tag{5.2}$$

$$\sigma^2(r_p) = \sum_{i=1}^{n+1} \sum_{j=1}^{n+1} w_i w_j \sigma_{ij} = \sum_{i=1}^{n+1} w_i^2 \sigma_i^2 + \sum_{i} \sum_{i \neq j} w_i w_j \sigma_{ij} \tag{5.3}$$

Für die Gewichtungen gilt:

$$\sum_{i=1}^{n+1} w_i = 1, \quad w_i \geq 0 \tag{5.4}$$

Die beiden Ziele: Maximierung des erwarteten Ertrags und Minimierung des Risikos werden in folgender Zielfunktion (utility function) ausgedrückt, indem die Formeln 5.2 und 5.3 in Zusammenhang gesetzt werden:

$$U = E(r) - \frac{1}{2} A \sigma^2(r) \tag{5.5}$$

Der Gewichtungsfaktor A drückt die Risiko-Präferenz oder -Aversion des Anlegers aus. Falls der Anleger Risiko vermeiden möchte, wählt er $A < 0$, falls er hingegen risikofreudig ist, wird $A > 0$ gewählt. Ist der Anleger neutral zum Risiko eingestellt, wird $A = 0$ gesetzt. Der repräsentative Investor wählt einen Faktor zwischen $2,0$ und $4,0$.

Um nun das optimale Portfolio zu finden, wird die obige Zielfunktion (5.5) bzgl. der Gewichtungen der einzelnen Wertpapiere maximiert, wobei weiterhin 5.4 gelten muss:

$$\max_{w_i} U = E(r) - \frac{1}{2} A \sigma^2(r) \tag{5.6}$$

Die Lösung dieser Maximierung besteht also aus einem Vektor $\overrightarrow{w_i}$ ($1 \leq i \leq n+1$), der zugleich die Lösung der Optimierung beschreibt und somit angibt, welche Bestandteile mit welchem Anteil in das Portfolio eingehen. Die Auswahl des optimalen Portfolios (5.6) kann durch ein *quadratisches Programm* geschehen.

5.3.1 Graphische Darstellung

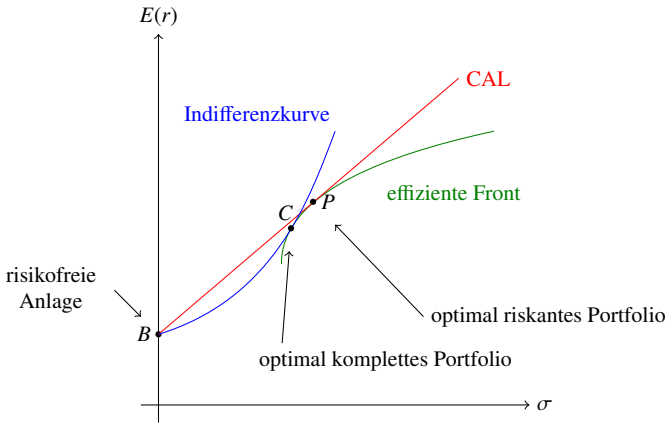


Abb. 5.2: Portfolioauswahl mit einer risikofreien Anlage

In Abb. 5.2 wird die Auswahl eines für einen Anleger optimalen Portfolios dargestellt. Grundlage dazu ist die effiziente Front, die in Abschnitt 5.2 mit Hilfe von Abb. 5.1 hergeleitet wurde. Des Weiteren wird eine risikofreie Anlage B benötigt. Diese beschreibt eine theoretische Ertragsrate bei $\sigma = 0$. Sie kann als der erwartete Zinssatz des Investors interpretiert werden. In der Praxis werden für die risikofreie Anlage solche Anlagen gewählt, die als sehr sicher gelten; z. B. Staatsanleihen. Wird von diesem Punkt B eine Tangente zur effizienten Front eingezeichnet, so bildet diese die *Capital Allocation Line (CAL)*, welche die Ertrag-zu-Risiko-Rate angibt. Auf den Schnittpunkt P der Tangente befindet sich das *optimal riskante Portfolio*. Es hat die beste Ertrag-zu-Risiko-Rate.

Im zweiten Schritt wird das für den Anleger optimale Portfolio gewählt. Dazu wird seine Risikofreudigkeit durch eine *Indifferenzkurve* modelliert. Diese kann sich also je nach Anleger unterscheiden (s. Abb. 5.3) [8]. Ausgangspunkt der Indifferenzkurve ist die risikofreie Anlage oder der vom Anleger erwartete Mindestzinssatz.

Jedes Portfolio, welches oberhalb der Kurve liegt, wäre dem Anleger recht. Damit nun aber das beste ausgewählt werden kann, wird die Indifferenzkurve so auf der Y-Achse verschoben, dass sie die effiziente Front tangential schneidet. Dies bedeutet formal, dass der Anleger seinen erwarteten Null-Risiko-Ertrag ändert. Der Schnittpunkt C ist dann das *optimal komplette Portfolio*.

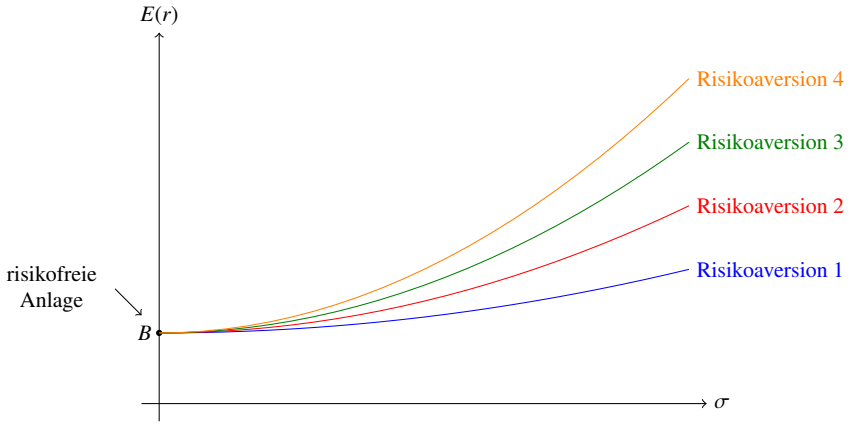


Abb. 5.3: Indifferenzkurven für verschiedene Risikotoleranzen

5.4 Elektrizitätsmarkt

Während traditionell eher große, alleinstehende Energieanlagen genutzt werden, von denen einfach die günstigste ausgewählt werden kann, gibt es auf dem heutigen Energiemarkt insbesondere durch die größere Verbreitung von erneuerbaren Energien viele kleine Kraftwerke. Dazu zählen beispielsweise Blockheizkraftwerke, Photovoltaik- und Windkraftanlagen. Da diese für sich genommen, weder genug Energie noch eine ausreichende Sicherheit für einen Börsenhandel bieten können, müssen sie in einem Portfolio zusammengefasst werden. Die beste Kombination von Kraftwerken kann mit der Portfoliooptimierung herausgefunden werden.

In dem Paper „An Optimization Approach to Select Portfolios of Electricity Generation Projects with Renewable Energies“ [4] wird der portugiesische Strommarkt zwischen 2009 und 2011 untersucht. Für die Übertragung der Portfoliooptimierung auf den Elektrizitätsmarkt werden lediglich die zuvor betrachteten Wertpapiere durch Elektrizitätserzeuger ersetzt. Der erwartete Ertrag wird zur erwarteten Energie, die in MWh angegeben wird. Einen Risikofaktor gibt es beim Elektrizitätsmarkt vor allem durch die wetterabhängigen Windkraft- und Photovoltaikanlagen. Bei erneuerbaren Energien wird der Nutzungsgrad (engl. capacity factor), der die Rate zwischen der tatsächlichen Energieproduktion und der theoretisch möglichen angibt, zur Berechnung des Risikofaktors genutzt. Dies ist dadurch begründet, dass bei erneuerbaren Energien die Anschaffungskosten sehr hoch sind, während im Vergleich die laufenden Betriebskosten nur einen geringen Teil ausmachen.

Die Stromerzeugungskosten (*levelized cost of electricity (LCOE)*) werden für die verschiedenen Energieerzeuger über den Zeitraum t abgeschätzt. $LCOE_t$ gibt die Gesamtkosten pro in der gesamten Lebenszeit produzierten MWh an:

$$LCOE_t = \frac{I + [M + (F_t + X_t)h]\frac{(1+r)^n - 1}{r(1+r)^n}}{E_t h \frac{(1+r)^n - 1}{r(1+r)^n}} \tag{5.7}$$

Dabei haben die Variablen die folgenden Bedeutungen:

I	Investitionskosten für jede Technologie
M	jährliche Betriebs- und Instandhaltungskosten
F_t	Treibstoffkosten
X_t	Umweltkosten
n	Lebensdauer der Anlage
E_t	Energieausgang, gemessen für jede Zeitperiode
t	Zeitperiode der Untersuchung in Stunden
h	Anzahl Stunden eines Jahres
r	Diskontsatz

Die Quellen für diese Parameter können in dem Paper [4, S. 359] nachgelesen werden.

Mit der Portfoliooptimierung kann nun die beste Zusammensetzung verschiedener Energieerzeuger herausgefunden werden. Der erwartete Ertrag einer einzelnen Anlage in einem Zeitintervall t wird durch das Inverse der Energieerzeugungskosten beschrieben:

$$r_{i,t} = \frac{1}{LCOE_t} \tag{5.8}$$

Der erwartete Ertrag für das Portfolio (r_p) kann mit (5.2) berechnet werden. Das Risiko wird mit Hilfe der ebenfalls schon beschriebenen Formel (5.3) berücksichtigt. Um diese nutzen zu können wurden in dem Paper die Kovarianzen zwischen den verschiedenen Technologien bestimmt. Die genauen Werte für die Kovarianzen und die daraus resultierenden Ergebnisse können in dem Paper [4] nachgelesen werden.

5.5 Kritik von Taleb

Kritik an der Portfoliooptimierung nach Markowitz und insbesondere der dort enthaltenen Normalverteilungsannahme bzgl. des Risikos wird von Nassim Nicholas Taleb geübt. Er ist Publizist und Börsenhändler, hat einen Doktor in Betriebswissenschaft und ist aktuell ein angesehener Professor für Risikoforschung am Polytechnischen Institut der Universität von New York. [12]

5.5.1 Der Schwarze Schwan

Die Normalverteilungsannahme führt nach Talebs Meinung zur Unterschätzung des Risikos. Zur Verdeutlichung wird in Abb. 5.4 eine Normalverteilung gegeben. Darin

wurden die 50 und 90 % großen Bereiche gekennzeichnet, um zu veranschaulichen, wie wenig Einfluss die stärker abweichenden „Ereignisse“ haben. [7]

Für ein seltenes, aber nicht prognostizierbares Ereignis nennt Taleb in seinem Buch „Der Schwarze Schwan“ eben diesen als Beispiel. Die Redewendung *Schwarzer Schwan* hat ihren Ursprung im Lateinischen und galt noch im 16. Jahrhundert in London als übliche Bezeichnung einer unmöglichen Aussage, da bis dato stets nur weiße Schwäne bekannt waren und die Annahme, dass es auch schwarze Schwäne gäbe, absurd schien. Im Jahr 1967 wurden jedoch ebensolche schwarzen Schwäne in Australien entdeckt. Daher wird der Ausdruck seitdem als Bezeichnung für eine vermeintliche Unmöglichkeit verwendet. Die Falsifikation einer solchen als Prämisse genutzten Aussage führt auch zur Ungültigkeit all ihrer Konklusionen. [11]

Taleb fasst den Begriff mit drei Punkten zusammen: Seltenheit, extreme Auswirkung und im Nachhinein (jedoch nicht im Voraus) berechenbar. Im Bezug auf die Normalverteilungsannahme wäre ein Schwarzer Schwan also ein von dem Erwartungswert stark abweichender Fall, welchem dadurch eine sehr geringe Gewichtung zugemessen wird. Dies führt insbesondere dann zur Unterschätzung eines Randfalles, wenn dieser große Auswirkungen hat. [10]

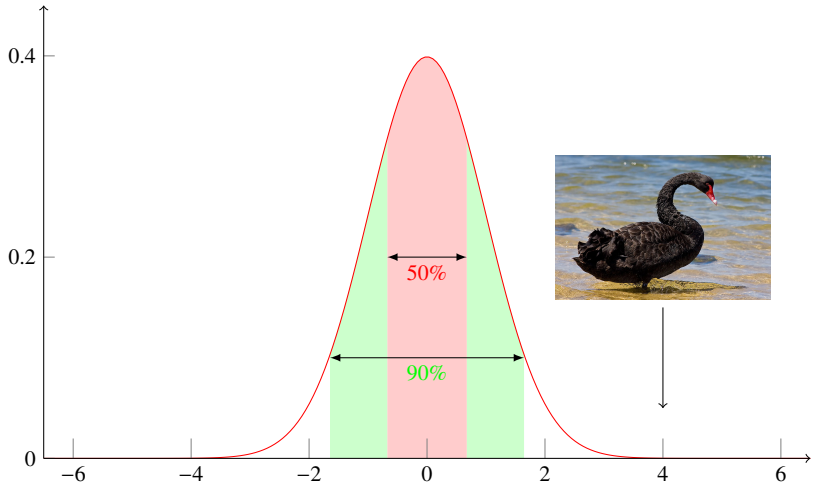


Abb. 5.4: Normalverteilung

Im Bereich der Energieerzeugung könnte z. B. ein Offshore-Windpark betrachtet werden. Im Normalfall würde eventuell jede Windkraftanlage mit einer Ausfallswahrscheinlichkeit versehen und so berücksichtigt werden. Allerdings würde der Ausfall einer Kabelverbindung zum Festland die Energieübertragung von allen Anlagen zugleich unmöglich machen. Dies ist genau ein solch seltener Fall, der sehr große Auswirkungen hat.

Im Bereich der Windenergie ist Prof. Peinke von der Carl von Ossietzky Universität in Oldenburg auf einen ähnlichen Schluss bzgl. der Problematik mit der Nor-

malverteilungsannahme gekommen. In seinem Paper „Fat tail statistics and beyond“ schreibt er, dass die Gauß-Statistik für die Abschätzung von Windkraftanlagen nicht geeignet ist, da bei solchen eine sog. *Fat-* bzw. *heavy-tailed distribution* auftritt. [5] Diese Namensgebung basiert auf dem Aussehen der Verteilungskurve, welche im Gegensatz zur Gauß-Kurve, die exponentiell nach außen hin abfällt, nur langsam abfällt.

5.5.2 Antifragilität

Taleb empfiehlt wegen der zuvor beschriebenen Problematik seltener Ereignisse in seinem Buch [10] fragile Finanzinstrumente zu meiden. Stattdessen sollten *antifragile* Instrumente bevorzugt werden. Diese Begriffe beschreiben das Verhalten von verschiedenen Gegenständen, Organismen oder Systemen bei äußeren Einflüssen. In Abb. 5.5 werden sie anhand von Beispielen auf einer „Auswirkungsachse“ eingeordnet.

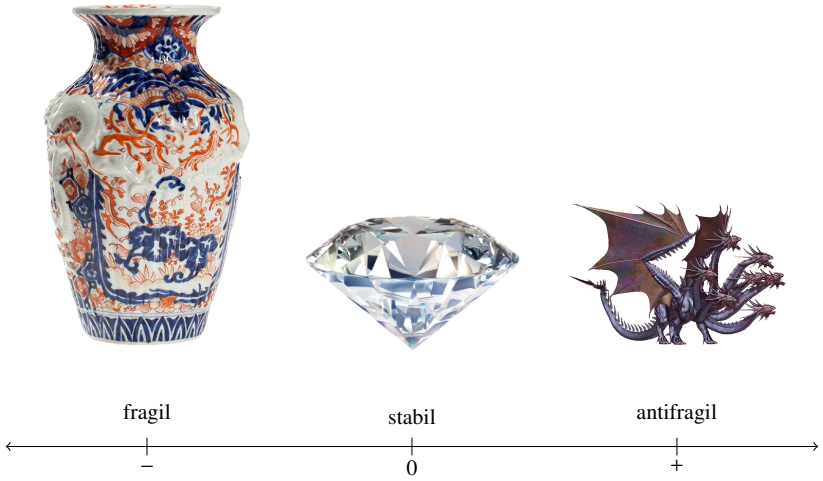


Abb. 5.5: Verhalten bei äußeren Einflüssen

Die Vase ist fragil, da sie bei äußerlicher Beschädigung leicht zerbricht, d. h. der äußere Einfluss hat negative Auswirkungen. Ein Diamant hingegen ist stabil, da er gegenüber äußeren Einflüssen (bis zu einem gewissen Maße) neutral gegenübersteht. Als Beispiel für Antifragilität dient hier die Hydra aus der griechischen Mythologie. Dieser Kreatur wachsen immer 2 Köpfe nach, sobald einer abgeschlagen wurde. Das bedeutet, ein äußerer Einfluss führt zu einer Verbesserung. In der Sage wurde das Ungeheuer schließlich von Herakles besiegt, indem die Köpfe nach dem Abschlagen

ausgebrannt wurden. Dies zeigt, dass die Antifragilität nur bis zu einem bestimmten Grad möglich ist.

Als reale Beispiele für die Antifragilität können der Muskelaufbau oder eine Populationsentwicklung genannt werden. Ein Muskel baut sich auf, wenn einzelne winzige Fasern reißen. Bei einer Überanstrengung kann es allerdings zum Muskelfaserriss kommen. Eine Population ist antifragil, da bei äußeren Einflüssen die schwachen Individuen herausfallen und die stärkeren überleben. An der Populationsentwicklung kann ein weiterer Aspekt verdeutlicht werden: Die Antifragilität ist nur dann möglich, wenn die Elemente (hier Individuen) der darunterliegenden Ebene fragil sind. Denn in diesem Beispiel müssen einzelne Individuen sterben, um die Population zu verbessern und zu erhalten. [9]

Betrachtet man nun wieder ein Portfolio, so fällt auf, dass die Voraussetzungen für den zuletzt genannten Aspekt schon gegeben sind, denn ein Portfolio kann aus vielen (möglicherweise fragilen) Wertpapieren bestehen, welche die darunterliegende Ebene bilden. Dadurch, dass nicht verlangt wird, dass jedes Wertpapier absolut sicher (stabil) ist, kann ein höheres Risiko eingegangen und so der mögliche Gewinn maximiert werden. Ein absolut sicheres Wertpapier könnte nur eine geringe Rendite erbringen.

Noch interessanter ist die (Anti-)Fragilität jedoch als Ersatz für abgeschätzte Risiken. Denn während Risiken nur abgeschätzt werden können, da hierbei eine Prognose für die Zukunft gemacht werden muss, und Wahrscheinlichkeiten für Erschütterungen oder Schwarze Schwäne nicht kalkulierbar sind, kann (Anti-)Fragilität gemessen werden. Daher sind Vergleiche verschiedener Objekte hinsichtlich ihrer Fragilität wesentlich genauer als Risikovergleiche. Es ist schwierig oder unmöglich festzustellen, ob ein bestimmtes seltenes Ereignis wahrscheinlicher ist als ein anderes. Jedoch lässt sich die Fragilität zweier Objekte beim Eintreten eines bestimmten Ereignisses recht gut vergleichen. So kann bspw. festgestellt werden, dass eine Bank, im Falle einer Krise, fragiler ist als eine andere, obwohl die Wahrscheinlichkeit für eine solche Krise unbekannt und sehr gering ist. [9]

5.6 Zusammenfassung

Die Portfoliotheorie nach Markowitz untersucht, wie optimale Portfolios aus risikobehafteten Wertpapieren gebildet werden können. Dazu werden die beiden Ziele verfolgt, den *erwarteten Ertrag* zu maximieren und das *Risiko* zu minimieren. Durch Anwendung der Portfoliooptimierung lässt sich die effiziente Front erstellen und daraus, je nach Risikopräferenz des Anlegers, ein für diesen passendes Portfolio auswählen. Eine Übertragung der Portfoliooptimierung auf den Elektrizitätsmarkt ist möglich, indem die einzelnen Wertpapiere durch Energieerzeuger ersetzt werden, welche einen Zusammenschluss in Form eines virtuellen Kraftwerks bilden sollen. Die Optimierung wählt dann die am besten geeigneten Energieerzeuger aus. Da die Portfoliotheorie auf vielen, teilweise unrealistischen Annahmen basiert, gibt es einige Kritikpunkte. Insbesondere wird die Normalverteilungsannahme aufgrund ihrer

Unterschätzung seltener Ereignisse kritisch gesehen. Diesen als *Schwarze Schwäne* bezeichneten Ereignissen kann durch die Bevorzugung von Antifragilität entgegengewirkt werden. Es gibt mehrere Erweiterungen, die einige Nachteile ausbessern. Dazu zählen u. a. die *Post-modern portfolio theory* und das *Black-Litterman-Verfahren*.

Literaturverzeichnis

[1] David Gerginov. *Vollkommener Kapitalmarkt – Erläuterung einer Finanztheorie*. letzter Zugriff 23.06.2015. Juni 2013. URL: http://www.gevestor.de/details/vollkommener-kapitalmarkt-erlauterung-einer-finanztheorie-650795.html.

[2] Min Liu und Felix F. Wu. „Portfolio optimization in electricity markets“. In: *Electric Power Systems Research* 77 (2007), S. 1000–1009. URL: http://tinyurl.com/ng2seut.

[3] Harry M. Markowitz. „Foundations of Portfolio Theory.“ In: *Journal of Finance* 46.2 (1991), S. 469–477. ISSN: 00221082. URL: http://search.ebscohost.com.proxy01.bis.uni-oldenburg.de/login.aspx?direct=true&db=buh&AN=4653212&site=ehost-live.

[4] Eduardo Matos u. a. „An Optimization Approach to Select Portfolios of Electricity Generation Projects with Renewable Energies“. In: *Applied Mathematics & Information Sciences* 9 (2015), S. 357–363. URL: http://tinyurl.com/lacb2d9.

[5] J. Peinke u. a. „Fail Tail Statistics and Beyond“. In: *Advances in Solid State Physics* 44 (2004), S. 363–374. URL: http://www.uni-oldenburg.de/physik/forschung/twist/publikationen.

[6] MARK RUBINSTEIN. „Markowitz's „Portfolio Selection“: A Fifty-Year Retrospective.“ In: *Journal of Finance* 57.3 (2002), S. 1041–1045. ISSN: 00221082. URL: http://search.ebscohost.com.proxy01.bis.uni-oldenburg.de/login.aspx?direct=true&db=buh&AN=6778484&site=ehost-live.

[7] Frank Schmielewski. „Schwarze Schwäne, Antifragilität und Portfoliooptimierung“. In: *Risiko Manager* 20 (Sep. 2012). URL: https://www-wiso-net-de.proxy01.bis.uni-oldenburg.de/toc_list/RISK/2012/DT=20120927/Heft+20+%2F+2012/RISK#documentLayer.RISK__20120927bankv_rm_1220001.

[8] William C. Spaulding. *Modern Portfolio Theory: Efficient and Optimal Portfolios*. letzter Zugriff 21.06.2015. 2014. URL: http://thismatter.com/money/investments/modern-portfolio-theory.htm.

[9] Nassim Nicholas Taleb. *Antifragilität: Anleitung für eine Welt, die wir nicht verstehen*. ISBN: 3813504891. München : Knaus, 2013.

[10] Nassim Nicholas Taleb. *Der Schwarze Schwan - Die Macht höchst unwahrscheinlicher Ereignisse*. Carl Hanser Verlag GmbH & Co. KG, 2008. DOI: 10.3139/9783446419377.

[11] Wikipedia. *Black swan theory*. Wikipedia, The Free Encyclopedia. letzter Zugriff 21.06.2015. 2015. URL: https://en.wikipedia.org/w/index.php?title=Black_swan_theory&oldid=667424368.

[12] Wikipedia. *Nassim Nicholas Taleb*. Wikipedia, The Free Encyclopedia. letzter Zugriff 21.06.2015. 2015. URL: https://en.wikipedia.org/w/index.php?title=Nassim_Nicholas_Taleb&oldid=665321538.

Kapitel 6
Einsatzplanoptimierung für virtuelle Kraftwerke

Stefanie Holly

Zusammenfassung Die Einsatzplanung ist eine von Energieversorgungsunternehmen durchgeführte Tätigkeit, um Lieferverpflichtungen durch Produktion von elektrischer Energie effizient erfüllen zu können [10]. Ein virtuelles Kraftwerk ist ein Verbund vieler dezentraler Energieerzeuger. Das Ziel der Einsatzplanung ist es, die Fahrpläne der einzelnen Erzeuger so aufeinander abzustimmen, dass die gewünschte Leistung durch das virtuelle Kraftwerk bereitgestellt wird. Im Rahmen dieser Arbeit werden zunächst die Grundlagen der Einsatzplanung vermittelt. Es erfolgt eine Einteilung in prädiktive und reaktive Einsatzplanung, sowie eine weitere Unterteilung der prädiktiven Planung in langfristige und kurzfristige Planungsprozesse. Daraufhin wird die zentrale Problemstellung in Form des „Unit Commitment Problem“ und „Economic Dispatch Problem“ erläutert und anschließend die Besonderheiten im Kontext von virtuellen Kraftwerken erörtert. Des Weiteren werden verschiedene Lösungsalternativen vorgestellt. Dabei handelt es sich um mathematische Verfahren, evolutionäre Algorithmen und Multiagentensysteme. Abschließend werden die verschiedenen Aspekte zentraler und dezentraler Kontrollstrukturen betrachtet. Die Auswahl geeigneter Kontrollstrukturen für das Projekt „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ kann in dieser Arbeit nicht getroffen werden, da sie von der noch ausstehenden Anforderungsanalyse abhängig ist.

6.1 Einleitung

Im März 2015 war Deutschland der Schauplatz einer partiellen Sonnenfinsternis. Bereits im Vorfeld wurde darüber spekuliert, ob es den Netzbetreibern gelingen würde, die extremen Schwankungen erzeugter Solarenergie auszugleichen. Die Vorbereitung der Versorger für diese kritische Situation nahm ein ganzes Jahr in Anspruch. [5] Der Grund aus dem dieses Ereignis eine solch große Herausforderung darstellte,

Carl von Ossietzky Universität Oldenburg
E-mail: stefanie.holly@uni-oldenburg.de

ist der plötzliche Wegfall enormer Kapazitäten zu Beginn der Sonnenfinsternis und der sprunghafte Leistungsanstieg bei ihrem Ende. Um diese Schwankungen auszugleichen, mussten die Versorger diese Veränderungen genau prognostizieren und durch das Zuschalten bzw. Abkoppeln anderer Kraftwerke ausgleichen. Ein solcher Planungsprozess wird als Einsatzplanung bezeichnet. Die Einsatzplanung erfolgt nicht nur bei außergewöhnlichen Ereignissen wie einer Sonnenfinsternis, sondern wird für jeden Tag des Jahres durchgeführt.

Im Rahmen dieser Arbeit werden zunächst die Grundlagen der Einsatzplanoptimierung erläutert. Dabei werden die unterschiedlichen Arten der Einsatzplanung beschrieben. Daraufhin wird das zu lösende Optimierungsproblem konkretisiert und auch eine mathematische Formulierung dargelegt. Anschließend folgt eine Konkretisierung der allgemeinen Problemstellung auf den Kontext virtueller Kraftwerke. Im Fokus dieses Kapitels steht die Betrachtung verschiedener Lösungsalternativen. Dabei werden sowohl mathematische als auch heuristische Verfahren angeführt. Bei den beiden ausgewählten heuristischen Verfahren handelt es sich um evolutionäre Algorithmen und Multiagentensysteme. Daraufhin werden verschiedene Organisationsstrukturen vorgestellt und ein abschließendes Fazit gegeben.

6.2 Grundlagen der Einsatzplanoptimierung

Das Hauptziel der Einsatzplanung in elektrischen Systemen ist die Menge erzeugter Energie genau auf die Nachfrage abzustimmen. Es müssen eine zuverlässige Versorgung gewährleistet, zugleich die anfallenden Kosten minimiert und eine Reihe von Einschränkungen eingehalten werden [Seite 1][3].

Die Einsatzplanung in der Elektrizitätswirtschaft lässt sich grundsätzlich in zwei Arten unterteilen. Diese sind die prädiktive und die reaktive Einsatzplanung. Die Differenzierung erfolgt anhand des Planungszeitpunktes. In Abbildung 6.1 ist zu erkennen, dass der prädiktive Planungsprozess vor dem Planungshorizont erfolgt. Der Planungshorizont beschreibt hierbei einen bestimmten Zeitraum, für den die Einsatzplanung durchgeführt wird. Der Einsatzplanungsprozess ist eine Abfolge mehrerer aufeinanderfolgender Planungsphasen, wie in Abbildung 6.2 veranschaulicht wird. Es beginnt mit strategischer und langfristiger Planung, die Jahre oder Monate im Voraus erfolgt. Diese Planung kann beispielsweise Lieferverträge für Rohstoffe wie Kohle oder Gas beinhalten. Darauf wird die Planung für immer kleiner werdende Zeitintervalle wiederholt, bis zuletzt die Tagesplanung „in der Regel Day-Ahead, d.h. bis zu 24 Stunden vor einer tatsächlichen Bereitstellung von Elektroenergie“ durchgeführt wird [9, Seite 23].

Die Bezeichnung „prädiktiv“ impliziert bereits, dass diese Planung auf Vorhersagen und Prognosen beruht. Es existieren jedoch auch unvorhersehbare Ereignisse, die zu Störungen des elektrischen Systems führen können. Dabei kann es sich beispielsweise um durch einen Sturm verursachte Leitungsschäden oder einen Störfall in einem Kraftwerk handeln. Im Kontext von erneuerbaren Energien spielt auch die Wettervorhersage eine entscheidende Rolle um die durch Windkraft- oder Photovol-

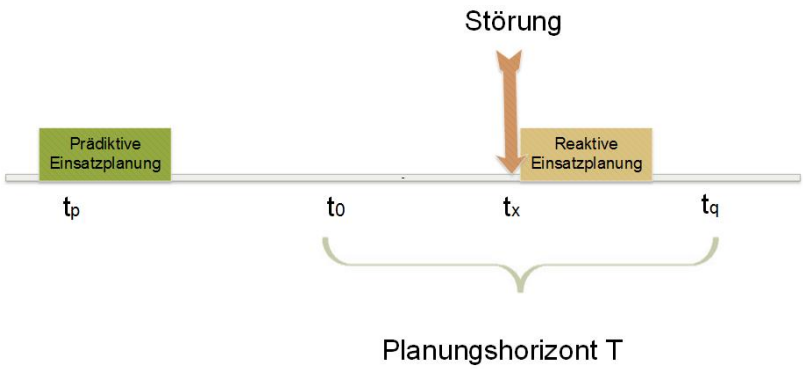


Abb. 6.1: Zeitpunkt der Einsatzplanung nach [2, Seite 73]

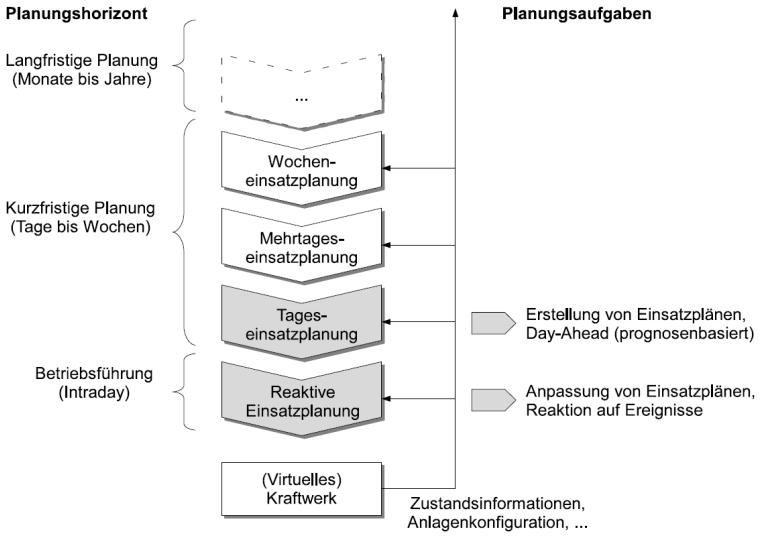


Abb. 6.2: Einsatzplanung als betriebswirtschaftliche Prozesskette mit aufeinander aufbauenden Planungsprozessen [9, Seite 23]

taikanlagen erzeugte Energie korrekt prognostizieren zu können. Diese Vorhersagen sind nicht immer voll und ganz zutreffend. Tritt ein unvorhergesehenes Ereignis ein, müssen ausgleichende Maßnahmen ergriffen werden. Dies geschieht im Rahmen der reaktiven Einsatzplanung.

Für die Projektgruppe „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ ist die kurzfristige prädiktive Einsatzplanung von Interesse. Dennoch soll in den folgenden beiden Unterkapiteln zunächst auf die langfristige prädiktive und auf die reaktive Einsatzplanung eingegangen werden. Dies dient der vollständigen Beschreibung aller Facetten der Einsatzplanung. In Abschnitt 6.3 wird ausführlich auf die Problematik der kurzfristigen prädiktiven Einsatzplanung eingegangen.

6.2.1 Prädiktive Einsatzplanung

In Abbildung 6.2 war bereits zu erkennen, dass die prädiktive Einsatzplanung sowohl langfristige als auch kurzfristige Planung umfasst. In diesem Kapitel soll auf die langfristige prädiktive Einsatzplanung eingegangen werden.

Die konventionelle und langfristige Einsatzplanoptimierung umfasst die Planung von Energieerzeugung und Erhaltung für Monate, Jahre oder Jahrzehnte im Voraus. Sie dient in erster Linie dazu, die ausreichende Versorgung der Verbraucher zu gewährleisten. [3] Die Schwierigkeit der Aufgabe liegt zum einen in der Komplexität der elektrischen Systeme als solche begründet. Diese sind stark vermaschte Netzwerke, welche zahlreiche elektrische Maschinen und Anlagen, sowie Leitungen umfassen. Zum anderen müssen auch bei sich schnell ändernden Gegebenheiten zulässige Spannung und Frequenz erhalten werden.

Abbildung 6.3 zeigt die Aufgaben der Einsatzplanung und deren Beziehungen untereinander nach Hobbs [3]. Wie bereits angeführt wurde, handelt es sich um langfristige Planung, die mehrere Dekaden beinhaltet. Demzufolge stehen zu Beginn Prognosen zur demographischen und wirtschaftlichen Entwicklung des betrachteten Gebietes (1). Auf Basis dieser Prognosen ist es möglich, Vorhersagen über die zukünftige Entwicklung des Energiebedarfs zu treffen (2). Zugleich werden aus einer Fülle von Alternativen sowohl auf Angebots- als auch auf Nachfrageseite zahlreiche „Resourcenmixe“ erzeugt (3)(4). Auf der Angebotsseite betrifft dies die Auswahl unterschiedlicher Kraftwerksarten und Technologien. Dabei sind Faktoren, wie Investitionskosten, Betriebs- und Wartungskosten, sowie Betriebsmerkmale zu beachten. Zugleich existieren auch auf Verbraucherseite viele Ausgestaltungsmöglichkeiten, wie verbesserte Energieeffizienz, Energiespeicherung vor Ort und die gezielte Förderung des Nachtstromverbrauchs [3, Seite 2]. Im “Angebot-/ Nachfrageintegrationsmodell“ werden ausgewählte Optionen auf Anbieter- und Verbraucherseite zu einem Portfolio kombiniert, welches den vorhergesagten Bedarf erfüllt. Dieses Portfolio ist auf die Optimierung eines ökonomischen Ziels ausgerichtet, wie beispielsweise die Minimierung der Kosten. Zusätzlich bestehen Einschränkungen, sodass die erforderliche Zuverlässigkeit des Systems gewährleistet werden kann (6). Für den selektierten “Resourcenmix“ werden anschließend die anfallenden Kosten ermittelt

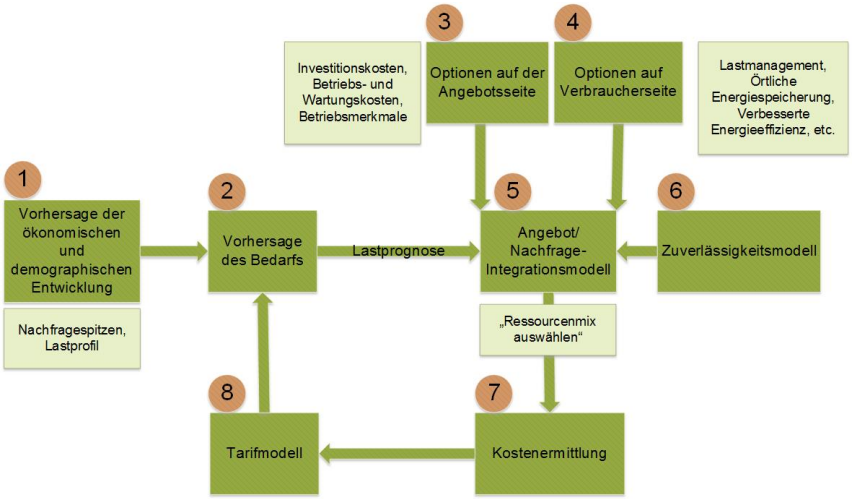


Abb. 6.3: Konventionelle Einsatzplanoptimierung nach [3, Seite 4]

(7) und auf Basis dieser das Tarifmodell angepasst (8). Veränderungen im Tarifmodell können sich auf die prognostizierte Nachfrage auswirken. Ist dies der Fall, muss die Auswahl eines geeigneten Portfolios gegebenenfalls erneut durchgeführt werden. [3, Seite 4]

Ein wesentlicher Komplexitätsfaktor der Einsatzplanoptimierung besteht in der Tatsache, dass neben der bedarfsangemessenen Versorgung meist die Erfüllung vieler weiterer Kriterien notwendig ist. Auf der einen Seite soll ein Optimierungsziel erfüllt werden, wie beispielsweise die bereits genannte Minimierung der Kosten. Auf der anderen Seite muss eine Reihe zusätzlicher Bedingungen eingehalten werden. Diese beinhalten die Erfüllung des erforderlichen Zuverlässigkeitsgrades, sowie die Einhaltung einiger technischer Einschränkungen (Auf diese wird in Abschnitt 6.3 näher eingegangen). Zugleich müssen auch ökologische und soziale Faktoren beachtet werden. Die Planung von Übertragungswegen geht beispielsweise häufig mit einer Abwägung zwischen anfallenden Kosten, Zuverlässigkeit, Flächennutzung, Ästhetik und möglicher Weise sogar gesundheitlichen Bedenken einher. [3, Seite 10] Die besondere Schwierigkeit besteht darin, bestehende Zielkonflikte aufzuzeigen und eine Priorisierung der unterschiedlichen Ziele durchzuführen. Nach Hobbs [3] kann die Gewichtung auf unterschiedliche Arten erfolgen. Eine Möglichkeit ist es, zunächst in Befragungen eine Priorisierung herzuleiten und diese auf verschiedene Alternativen anzuwenden. Gleichermaßen können auch mehrere Alternativen zur Auswahl stehen, welche dann bewertet und verbessert werden.

6.2.2 Reaktive Einsatzplanung

Das Produkt der prädiktiven Einsatzplanung ist ein Einsatzplan, der für alle beteiligten Anlagen den zu erzeugenden Leistungsverlauf vorgibt. Wie bereits zu Beginn von Abschnitt 6.2 erläutert wurde, müssen die Einspeisungen den Ausspeisungen zuzüglich der Verlustleistung entsprechen. Die Planung unterliegt jedoch einer gewissen Unsicherheit. Dies ist zum einen bereits durch die mathematischen Modelle, welche für die Prognose verwendet werden bedingt. Diese müssen teilweise stark vereinfacht werden, um berechenbar zu bleiben [9, Seite 25].

Abb. 6.4: Die Phasen der reaktiven Einsatzplanung nach [9]

Zum anderen wurde darauf hingewiesen, dass es auch zu unvorhergesehenen Störereignissen kommen kann. Unabhängig von der Ursache muss sowohl bei positiven als auch bei negativen Abweichungen schnell reagiert und mit Regelenergie ein Ausgleich geschaffen werden. Tröschel [9] definiert vier Phasen der reaktiven Einsatzplanung. Diese sind in Abbildung 6.4 dargestellt. Die erste Phase ist die Zustandsdatenerfassung. Dieser Prozess wird permanent ausgeführt und dient der Feststellung des aktuellen Systemzustands. Auf diese Weise können Abweichungen überhaupt entdeckt und infolgedessen auf sie reagiert werden. Auch die für eine Reaktion benötigten Daten können nur so erhoben werden. Darauf erfolgt die Anpassung des Fahrplans. Unter einem Fahrplan wird hierbei eine “zeitlich aufgelöste Leistungsvorgabe für einzelne Anlagen“ [9, Seite 32] verstanden. Fällt beispielsweise eine Anlage komplett aus, muss die für sie eingeplante Last auf andere Anlagen umverteilt werden. Anschließend kann eine Optimierung des Ersatzfahrplans erfolgen, um etwa die Kosten minimal zu halten. Abschließend werden die geänderten Fahrpläne an die betroffenen Anlagen übermittelt. Im weiteren Verlauf dieser Arbeit wird nur noch die prädiktive Einsatzplanung betrachtet, da sie für das betreffende Projekt die relevante Planungsaufgabe darstellt.

6.3 Unit Commitment und Economic Dispatch Problem

Das Unit Commitment Problem (UCP) und das Economic Dispatch Problem (EDP) sind für die Einsatzplanung in elektrischen Systemen essentiell. Durch die Lösung dieser beiden Probleme können Energieversorgungsunternehmen Einsatzpläne, die für ein bestimmtes Ziel optimiert sind und alle vorliegenden Einschränkungen berücksichtigen, erstellen.

Das UCP ist das Problem für eine Planungsperiode eine Auswahl von Energieerzeugern und deren Laufzeiten zu treffen. Dabei ist das Hauptziel die benötigte Systemleistung und erforderliche Regelleistung bereitzustellen. Dies soll mit minimalem Kostenaufwand erfolgen und unterliegt zudem einer Reihe weiterer Constraints (Einschränkungen). [7, Seite 185] Diese Einschränkungen sind in der konventionellen Einsatzplanung meist technischer Natur und entstehen durch spezifische Eigenschaften der einzelnen Anlagen. Unter konventioneller Einsatzplanung wird hierbei die Planung mit großen steuerbaren Kraftwerken verstanden, im Gegensatz zur Planung mit teilweise nicht steuer- oder regelbaren Erzeugern, wie Wind- oder Photovoltaikanlagen. Zu den Constraints zählen [7, Seite 188-190]:

- **Regelenergie**: Regelleistung muss zusätzlich zur Systemleistung bereitgestellt werden, um auf unerwartete Ereignisse schnell reagieren zu können. Die Vorhaltung der Regelenergie trägt erheblich zum Erreichen der erforderlichen Zuverlässigkeit des Systems bei.
- **Obere und untere Leistungsgrenzen**: Anlagen dürfen nur innerhalb dieser Leistungsgrenzen betrieben werden. Wird die Obergrenze überschritten, kann dies zu gefährlichen Überlastungen führen. Wird die Untergrenze unterschritten, ist dies in der Regel unwirtschaftlich.
- **Minimale Betriebs- und Ruhezeiten**: Laufende Anlagen dürfen erst nach einer bestimmten Mindestbetriebszeit wieder ausgeschaltet werden. Ebenso müssen ausgeschaltete Anlagen bestimmte Ruhezeiten nicht unterschreiten.
- **Anfangszustände der Anlagen**: Die Anfangszustände der Anlagen müssen berücksichtigt werden. Wurde eine Anlage beispielsweise vor dem Beginn der Planungsperiode längere Zeit nicht genutzt, hat dies Auswirkungen auf die erbrachte Leistung und die erzeugten Kosten für diesen Zeitraum.
- **Verfügbarkeit der Anlagen**: Die Verfügbarkeit der Anlagen kann zum Beispiel durch Wartungsarbeiten oder Ausfälle eingeschränkt sein.
- **Leistungsreduktion von Anlagen**: Der generelle Zustand der Anlagen muss gleichermaßen miteinbezogen werden. Alter oder veränderte Umwelt können sich auf obere und untere Leistungsschranken auswirken.

Während es das Ziel des UCP ist, geeignete Energieerzeuger und deren Laufzeiten auszuwählen, befasst sich das Economic Dispatch Problem damit, die Leistung optimal zwischen den laufenden Anlagen zu verteilen. Dabei müssen dieselben Einschränkungen wie beim UCP berücksichtigt werden. Ziel ist es auch hier die Gesamtkosten minimal zu halten. Das EDP ist somit ein Teilproblem des UCP. Es gilt aus der enormen Anzahl möglicher Fahrpläne der einzelnen Anlagen einen Gesamzfahrplan (Einsatzplan) zusammenzustellen, welcher alle geforderten Constrains und das Optimierungsziel erfüllt.

6.3.1 Mathematische Formulierung

Aus den Beschreibungen der beiden Probleme geht hervor, dass bei beiden eine Minimierung der Gesamtkosten angestrebt wird. Um diese zu erreichen, muss zunächst geklärt werden wie sich diese Kosten zusammensetzen. Gleichung 6.1 zeigt die Formel der Produktionskosten FC_i. Diese bezeichnet die Kosten, die direkt bei der Energieerzeugung entstehen. Die Kosten steigen quadratisch in Abhängigkeit der generierten Leistung P_i an. Bei A_i , B_i und C_i handelt es sich um anlagenspezifische Konstanten. [8] Besonders bei erneuerbaren Energien, wie Windkraft, bei denen keine Kosten für den Treibstoff anfallen, reduzieren sich diese Kosten erheblich.

$$FC_i(P_i) = (A_i P_i^2 + B_i P_i + C_i) \in /h \tag{6.1}$$

Da in den durch die Einsatzplanung generierten Fahrplänen Anlagen auch hoch und runtergefahren werden, müssen die „Start-Up“ und „Shut-Down“ -Kosten ebenfalls berücksichtigt werden. Die „Shut-Down“ Kosten SD_i sind in der Regel ein konstanter Faktor. Wie hoch die Startkosten SC_i einer Anlage ausfallen, ist von dem Zeitraum abhängig, in dem die Anlage zuvor nicht in Betrieb war. Übersteigt diese „Time Off“ (T_{off}) die „Unit cooling Time“ τ , also die Zeit nach der die Anlage vollständig ausgekühlt ist, gehen die kompletten Kaltstartkosten δ_i mit ein. Geht die „Time Off“ gegen 0 fallen nur die Kosten für eine „heiße Inbetriebnahme“ σ_i an. [8]

$$SC_i = \sigma_i + \delta_i * (1 - e^{\frac{-T_{off}}{\tau}}) \tag{6.2}$$

Die gesamten Kosten einer Anlage für einen Planungszeitraum setzen sich aus den Produktionskosten, Startkosten und „Shut-Down“ -Kosten zusammen, welche sich aus dem gewählten Fahrplan ergeben.

Diese Gesamtbetriebskosten müssen für jede Anlage berechnet und darauf die Kosten aller Einzelfahrpläne addiert werden, um die Gesamtkosten des Einsatzplans zu erhalten.

$$Minimize \sum_{t=1}^{T} \sum_{i=1}^{N} FC_{it} * U_{it} + SC_{it} * (1 - U_{i(t-1)}) * U_{it} + SD_{it} \tag{6.3}$$

Gleichung 6.3 illustriert die Zielfunktion des Optimierungsproblems. Das Ziel besteht darin, aus der Menge aller möglichen Fahrplankombinationen diejenige auszuwählen, welche diese Kostenfunktion minimiert und zugleich eine Reihe weiterer Bedingungen erfüllt. Die wichtigste Bedingung ist die Erzeugung der erforderlichen Leistung. Dazu kommen die in diesem Kapitel genannten technischen Einschränkungen, wie Leistungsgrenzen oder minimale Betriebszeiten von Anlagen. [8]

6.3.2 Anwendung auf virtuelle Kraftwerke

In diesem Kapitel wurde bisher die Einsatzplanung elektrische Systeme im Allgemeinen erörtert. Dieser Abschnitt dient der speziellen Betrachtung der Einsatzplanung in virtuellen Kraftwerken (VK). Die Problemstellung bleibt auch für VKs erhalten. Es gilt eine festgelegte Leistung zu erzeugen und dabei eine Zielfunktion zu optimieren (i.d.R. Kosten minimieren) und Constraints einzuhalten. Somit lassen sich UCP und EDP mit ihren Anforderungen und Einschränkungen komplett übertragen. Es kommen jedoch noch einige zu beachtende Faktoren hinzu (Aufzählung nach [10]):

- **Heterogene Anlagenstruktur**: VKs können viele verschiedene Kraftwerkstypen mit unterschiedlichen technischen und wirtschaftlichen Eigenschaften beinhalten. Einige davon sind nicht steuerbar, wie beispielsweise Photovoltaik- oder Windkraftanlagen. Andere sind zwar steuerbar, weisen jedoch Besonderheiten bei der Steuerung auf, die bei der Einsatzplanung zu beachten sind. Biogasanlagen besitzen beispielsweise eine sehr lange Anlaufzeit. „Vom Zeitpunkt des Einschaltens [...] bis zur tatsächlichen Energieproduktion können mehrere Tage vergehen“.
- **Höhere Anlagenanzahl**: Im Vergleich zu traditionellen Großkraftwerken erzeugen die kleinen dezentralen Anlagen erheblich geringere Produktionsleistungen. Um die gleiche Menge an elektrischer Energie zu generieren, ist daher eine weitaus größere Menge von Anlagen notwendig. Diese enorme Menge zu beachtender Anlagen erhöht die Komplexität des Einsatzplanungsproblems erheblich.
- **Produktionsunsicherheiten**: Die Leistung mancher Anlagen ist vom Wetter abhängig. (Bsp.: Photovoltaik und Windkraft). Daher kann die Leistung dieser Anlagen nur auf Grundlage der Wetterprognose vorhergesagt werden. Abweichungen der prognostizierten Leistung von der tatsächlich generierten können zu erheblichen Kosten führen, da fehlende Energiemengen kurzfristig beschafft werden müssen.
- **Steigender Wettbewerbsdruck**: Im Gegensatz zu den früheren monopolartigen Marktstrukturen existiert heute ein steigender Wettbewerbsdruck zwischen einer Vielzahl von Anbietern. Daher wird die Kostenminimierung ein immer wichtigerer Faktor bei der Einsatzplanung.
- **Unvollständige und ungenaue Informationen**: Durch die Dezentralisierung der Anlagen liegen in der Regel weniger oder ungenaue Daten über die aktuellen Zustände der Anlagen vor. Dafür existieren zwei Ursachen. Zum einen liegt aus Kostengründen oft keine geeignete Kommunikationsinfrastruktur vor. Zum anderen sind die Kosten für aufwendige Messeinrichtungen bei kleineren Anlagen oft unverhältnismäßig zu den Gesamtkosten der Anlagen, weshalb meist auf sie verzichtet wird.

Bei einem VK handelt es sich um eine Zusammenfassung von dezentralen Energiesystemen (Erzeuger, Verbraucher, Speicher) mit dem Ziel einer direkten Koordination auf der IKT-Ebene [9, Seite 19]. Winkels stellt in [10] ein Referenzmodell für die Tageseinsatzplanung dezentraler heterogener Energieerzeugungsanlagen vor. Er unterteilt dabei die Planung in vier Phasen. Diese sind in Abbildung 6.5 abgebildet. Zunächst erfolgt die Phase der Grunddatenerzeugung. Diese dient der Erfassung

aller für das Optimierungsproblem relevanten Daten. Zu diesem Zweck muss als erstes der Anlagenbestand ermittelt werden. Darauf werden die Einsatzrestriktionen der Anlagen hinzugefügt. Dabei handelt es sich um die zuvor bereits erwähnten Einschränkungen, wie Leistungsgrenzen oder minimalen Betriebszeiten. Zusätzlich werden die Anlagenkosten, also sowohl Start- als auch Betriebskosten für jede einzelne Anlage erfasst.

Abb. 6.5: Die vier Phasen der Tageseinsatzplanung nach dem Referenzmodell in [10]

In der zweiten Phase erfolgt die Bedarfsplanung. Durch Abschätzungen über den zukünftigen Bedarf oder durch Verpflichtungen aus bestehenden Verträgen wird die bereitzustellende Bruttolast (Lieferverpflichtung) ermittelt. Anschließend werden im Vorfeld getätigte Energieeinkäufe und die Energieerzeugung aus nicht-steuerbaren Systemen (z.Bsp. Windkraft- oder PV-Anlagen) berücksichtigt. Das Ergebnis ist die sog. Nettolast, welche es gilt durch die Einsatzplanung der steuerbaren Anlagen zu erzeugen. Genau dieser Planungsschritt erfolgt als Nächstes. In der Phase der Anlageneinsatzplanung findet die eigentliche Einsatzplanung, also die Lösung des UCP statt. Auf Grundlage der in den vorherigen Phasen ermittelten Parameter wird unter Verwendung einer geeigneten Lösungsmethode (siehe Abschnitt 6.4) ein möglichst optimaler Gesamtfahrplan erstellt. Abschließend werden in der Phase der Marktkommunikation die gewählten Fahrpläne sowohl an die einzelnen Anlagen als auch an die Netzbetreiber weitergeleitet. Dabei müssen rechtliche, technische und organisatorische Rahmenbedingungen der Energiewirtschaft berücksichtigt werden.

6.4 Lösungsalternativen

Grundsätzlich können Optimierungsprobleme durch mathematische oder heuristische Verfahren oder Kombinationen dieser gelöst werden. Mathematische Verfahren bieten den Vorteil, eine wirklich optimale Lösung zu finden. In der Praxis vorhandene Probleme sind jedoch oft von großer Komplexität und können nur durch Vereinfachung auf ein geeignetes mathematische Modell abgebildet werden. Heuristische Verfahren hingegen nähern die optimale Lösung Schritt für Schritt an. Sie können meist auch bei sehr komplexen Problemstellungen eingesetzt werden, garantieren aber nicht, dass Optimum zu finden. Im Folgenden werden verschiedene Lösungsmöglichkeiten für das hier vorliegende Optimierungsproblem betrachtet.

6.4.1 Mathematische Verfahren

Das UCP ist wie bereits in Abschnitt 6.3 erörtert wurde ein kombinatorisches Optimierungsproblem, mit einer Zielfunktion und einer Reihe von Randbedingungen, welche Einschränkungen aus der realen Welt abbilden und zugleich Entscheidungsvariablen darstellen. Durch die Einhaltung dieser Randbedingungen werden gültige Lösungen erzeugt. Im Weiteren soll das Beispiel 6.4.1(angelehnt an Beispiel 2.1 in [10]) dem Verständnis und dem Vergleich der Lösungsalternativen dienen. In dem Beispiel soll für zwei Gasturbinenkraftwerke K1 und K2 ein optimaler Einsatzplan gefunden werden, der alle geforderten Bedingungen erfüllt.

Eigenschaften	Kraftwerk K1	Kraftwerk K2
max. Produktionsmenge (in MWh)	6	7
CO_2 - Emmisionen (in kg/MWh)	450	560
$Erdgasverbrauch\ in\ m^3/MWh$	90	80
Produktionskosten (in €/MWh)	55	77

Abb. 6.6: Beispiel für Kraftwerksrestriktionen angelehnt an Beispiel 2.1 in [10]

Beispiel 6.4.1 *Lineare Optimierung*

In Abbildung 6.6 werden die technischen Eigenschaften der zwei Gasturbinenkraftwerke K1 und K2 dargestellt. Folgende Bedingungen sollen erfüllt werden:

- *Beide Kraftwerke sollen gemeinsam mindestens 10 MWh erzeugen*
- *Die Kosten sollen minimal gehalten werden*
- *Es sollen max. 5.240 kg* CO_2 *ausgestoßen werden*
- *Es stehen max. 888* m^3 *zur Verfügung*

Es gilt also für die beiden Variablen x_1 *und* x_2*, welche die Menge an produzierten MWh von K1 und K1 darstellen, eine optimale Belegung zu finden. Aus den Beschränkungen lässt sich folgendes lineares Ungleichungssystem ableiten:*

$$\begin{pmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ 450 & 560 \\ 90 & 80 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} -10 \\ 6 \\ 7 \\ 5.240 \\ 888 \end{pmatrix}$$

Die Zielfunktion sieht folgendermaßen aus:

$$min \longrightarrow F(x) := \begin{pmatrix} 55 & 77 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Um dieses Problem mathematisch zu lösen, gibt es verschiedene Techniken. Ein Beispiel ist das sog. Simplex-Verfahren. Dabei steigt der Aufwand mit der Anzahl an Ungleichungen und Dimensionen des Problems. Das Simplex-Verfahren dient der Lösung linearer Optimierungsprobleme. Die Einsatzplanung ist jedoch in der Regel kein lineares Problem, da die Leistung des Gesamtsystems sprunghaft ansteigen oder abfallen kann, wenn einzelne Anlagen an- oder abgeschaltet werden.

Die Anwendung analytischer Verfahren verlangt Stetigkeit und Differenzierbarkeit der Funktionen. Zudem sind sie nicht anwendbar, wenn nur Teile des Definitionsbereichs untersucht werden sollen. „Wegen des im Vergleich zu den heuristischen Verfahren wesentlich größeren zu durchsuchenden Bereiches benötigen die mathematischen Verfahren eine höhere Rechenzeit“ [1, Seite A5]. Auch wenn eine optimale Lösung gefunden wird, muss es sich nicht um die Optimale Lösung für die Praxis handeln. Dies hängt maßgeblich von der Genauigkeit des zugrundeliegenden Modells ab, „welches um berechenbar zu bleiben, stark vereinfacht werden muss “ [1], da es sonst schnell zu exponentiellem Rechenaufwand kommen kann. Daher sind mathematische Verfahren „für die Optimierung realer Energiesysteme nicht geeignet“ [1, Seite A5]

6.4.2 Heuristische Verfahren

„Der Begriff „Heuristik“ bedeutet soviel wie „Kunst des Entdeckens“ und beschreibt damit Verfahren, systematisch Probleme zu lösen“ [1]. Es handelt sich in der Regel um einfache Verfahren, die auf Erfahrung beruhen und versuchen die optimale Lösung Schritt für Schritt annähern. Der Rechenzeitbedarf ist meist geringer als bei mathematischen Verfahren. Es existiert allerdings keine Garantie dafür, das Optimum tatsächlich zu finden. In den folgenden Abschnitten sollen aus der Fülle heuristischer Methoden zwei exemplarisch betrachtet werden.

6.4.2.1 Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) orientieren sich, wie der Name impliziert am biologischen Prinzip der Evolution. Abbildung 6.7 zeigt den Ablauf von Evolutionären Algorithmen.

Nachdem die Zielfunktion für das zu lösende Problem bekannt ist, gilt es eine geeignete Codierung für einzelne Lösungsvarianten zu finden (1). Die so codierten „Chromosomen“ werden durch einen Vektor aus binären bzw. reellen Zahlen oder anderen Elementen beschrieben [1, Seite 38]. Bei dem Problem aus Beispiel 6.4.1 kann ein Individuum durch $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ charakterisiert werden, wobei x_1 und x_2 wieder die Menge produzierter Energie der beiden Kraftwerke darstellen. Nachdem eine sinnvolle Codierung gefunden wurde, wird die Startpopulation erzeugt (2). Für das Beispielszenario könnte dies eine Menge beliebig gewählter Tupel sein, bei denen

Abb. 6.7: Evolutionärer Algorithmus

die Produktionsmenge jeweils zwischen 0 und dem Maximum liegt. Wir nehmen folgende Menge als Startpopulation an: $P := \{(0,7),(3,5),(5,5),(4,6),(6,7)\}$ Der nächste Schritt ist die Bewertung der Fitness der einzelnen Individuen (3). Da in dem vorliegenden Beispiel die Kosten minimiert und gleichzeitig eine Reihe von zusätzlichen Bedingungen erfüllt werden müssen, eignet sich folgende Funktion als Fitnessfunktion:

$$F(x) := \begin{cases} \left[\begin{pmatrix} 55 & 77 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right]^{-1}, & \text{wenn x alle Bedingungen erfüllt} \\ 0, & \text{sonst} \end{cases}.$$

Alle Tupel bis auf (5, 5) und (4, 6) besitzen eine Fitness von 0, da sie die Randbedingungen nicht erfüllen. Daher würden nur diese zwei Tupel mit der größten Fitness für den darauffolgenden Rekombinationsprozess (4) verwendet werden. Die durch die Rekombination (5) erzeugten Kinder wären (5, 6) und (4, 5). In der Mutationsphase (6) werden „zufallsgesteuert einzelne Bausteine verändert“ [10]. Dies könnte in diesem Fall so aussehen, dass (5,6) zu (5,4) und (4,5) zu (6,5) wird. Anschließend wird die Fitness der Kindergeneration berechnet (7) und darauf die besten Individuen ausgewählt (8). Solange das Abbruchkriterium nicht erfüllt ist, werden die Schritte 3 bis 8 wiederholt. Da nicht im Voraus bestimmt werden kann, nach wie vielen Iterationen eine optimale Lösung gefunden wird, werden für Abbruchkriterien andere Faktoren berücksichtigt. Der Algorithmus stoppt beispielsweise nach einer bestimmten Rechenzeit oder wenn die Fitness sich von Generation zu Generation nur noch geringfügig verändert. Es kann auch ein bestimmter Fitnessgrad bestimmt werden, ab dem die Lösung ausreichend gut ist.

Liegen sehr viele Constraints vor, wird ein Optimierungsproblem für eine mathematische Lösung zu komplex (exponentielle Rechenzeit). Wohingegen der Rechenaufwand von evolutionären Algorithmen „annähernd linear mit der Größe und Komplexität des Problems“ steigt, wodurch „Optimierungsaufgaben sehr detailliert modelliert werden“ können [1, Seite A14]. Zudem sind EA besonders für multikriterielle Optimierungen geeignet, da der Algorithmus und das Modell der betrachteten Betriebsmittel, sowie die Fitnessfunktion strikt voneinander getrennt sind. [1]. Allgemein ist der Rechenzeitbedarf von EA im Vergleich zu anderen Methoden groß. Wesentliche Faktoren für die Konvergenzgeschwindigkeit sind eine geeignete Codierung des Problems und die Auswahl der Startpopulation. Es ist möglich durch Erfahrung die Startpopulation sinnvoll zusammenzustellen und dadurch die Laufzeit des Algorithmus extrem zu verkürzen. Je näher die Startpopulation an der tatsächlichen Lösung ist, desto schneller konvergiert der Algorithmus. Evolutionäre Algorithmen sind für die praktische Lösung des für die Projektgruppe relevanten Problems geeignet, da sie universell für Optimierungsprobleme einsetzbar sind. In der Praxis werden sie bereits alleine oder in der Kombination mit anderen Verfahren zur Einsatzplanung in konventionellen Energiesystemen eingesetzt [1, Seite A15]. Es muss jedoch bedacht werden, dass für die Verwendung von EA alle relevanten Daten an zentraler Stelle vorliegen müssen. Wie dies bei einem verteilten System, wie einem virtuellen Kraftwerk zu bewerten ist, wird in Unterabschnitt 6.4.3 untersucht.

6.4.2.2 Multiagentensysteme

Agenten stammen aus dem Bereich der künstlichen Intelligenz. Es existiert keine einheitliche Definition. Nach Russell und Norvig ist ein Agent: „‘[...] alles, was seine Umgebung über Sensoren wahrnehmen kann und in dieser Umgebung durch Aktuatoren handelt.“ [6] Ein Agent kann demnach Informationen über seine Umwelt wahrnehmen und auch selbst Handlungen ausführen. Etwas genauer fällt die Definition von Wooldridge aus: „Ein Agent ist ein Computersystem, das in einer Umgebung lokalisiert und in dieser fähig ist, autonome Aktionen auszuführen, um seine Zielvorgaben zu erreichen.“ [11] Eine wesentliche Eigenschaft eines Agenten ist laut dieser Definition die Autonomie. Damit ist es dem Agenten möglich selbstständig zu handeln. Diese Autonomie resultiert in drei essentiellen Eigenschaften:

- Reaktivität
- Proaktivität
- Soziale Fähigkeiten

Demzufolge kann ein Agent Veränderungen in der Umwelt nicht nur wahrnehmen, er kann auch auf sie reagieren. Zugleich kann der Agent auch ohne äußere Anstöße aktiv werden. Seine sozialen Fähigkeiten machen die Interaktion mit anderen Agenten möglich.[9, Seite 30]

Ein Multiagentensystem (MAS) ist ein Netzwerk von Agenten, die interagieren um ein Problem zu lösen. Die Agenten können gemeinsame Ziele (Kollaboration), individuelle Ziele oder eine Kombination von beidem verfolgen. Eine wesentliche Eigenschaft eines MAS ist die eingeschränkte Sicht der einzelnen Agenten. Keiner der Agenten besitzt hinreichendes Wissen für die alleinige Lösung des Problems.

Eine Lösungsmöglichkeit der Einsatzplanoptimierung mit dezentralen Akteuren mit Hilfe von Multiagentensystemen wird in [2] vorgeschlagen. Dabei wird jede Anlage durch einen Agenten repräsentiert. Das globale Ziel aller Agenten ist die Maximierung der globalen Güte, welche sich aus der möglichst genauen Reproduktion des geforderten Wirkleistungsprodukts ergibt. Es besteht die Möglichkeit zusätzlich ökonomische oder ökologische Kriterien miteinzubeziehen, indem diese in der Zielfunktion ebenfalls gewichtet werden. Zur Erreichung dieses Ziels kollaborieren die Agenten. Zusätzlich können die Agenten als sekundäre Optimierungsziele versuchen, lokale Präferenzen durchzusetzen. Durch diese Verfolgung mehrere Ziele können Konflikte zwischen inkompatiblen Teilzielen entstehen. Die Agenten stehen also auch im Wettbewerb zueinander. Wie egoistisch bzw. altruistisch sich die Agenten verhalten kann, an das konkrete Modellierungsproblem angepasst werden.

Es herrscht das Prinzip der reaktiven Koordinierung vor. Die Agenten besitzen somit ein Schwarmverhalten, was bedeutet, dass sie auf die veränderte Auswahl ihrer Nachbarn reagieren. Die Grundstruktur des komplett dezentralen Ansatzes ist ein vollvermaschtes Netz. Da die Änderung der Auswahl eines Agenten in einer solchen Topologie allen Nachbarn, und somit allen anderen Agenten mitgeteilt würde, wird ein sogenanntes Overlay-Netz verwendet, um einen Kommunikations-Overload zu verhindern. Durch das Overlay-Netz wird die Anzahl der Nachbarn für jeden Agenten stark reduziert.

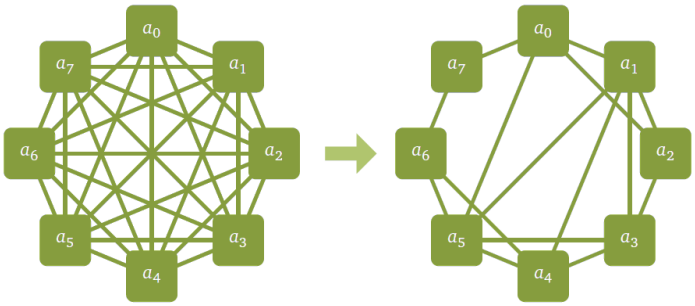


Abb. 6.8: Exemplarisches Overlay-Netz (rechts) für ein vollvermaschtes Kommunikationsnetz (links) mit acht Agenten aus [2]

Wie bereits oben erläutert, besitzen die Agenten eine eingeschränkte Sicht. Es existiert somit kein Agent, der alle Informationen besitzt. Um dennoch einen globalen Konsens zu erreichen, erfolgt eine Informationssynthese durch Anpassung der Nachrichten. Die Agenten versenden jedes Mal, wenn sich etwas in ihrem Arbeitsgedächtnis geändert hat, ihr komplettes Arbeitsgedächtnis. Dieses Gedächtnis umfasst:

- das zu erreichende Wirkleistungsprodukt
- den wahrgenommenen Systemzustand
- den Lösungskandidaten

Sowohl bei dem wahrgenommenen Systemzustand als auch beim Lösungskandidaten handelt es sich um eine Konfiguration des Gesamtsystems, also um einen möglichen Einsatzplan, der für jede Anlage genau einen Fahrplan enthält. Die Fahrpläne der einzelnen Agenten besitzen jeweils eine eigene Zählvariable. Erhält ein Agent eine Nachricht, so kann er zunächst seinen wahrgenommenen Systemzustand aktualisieren, indem er überprüft, ob Fahrplanauswahlen mit neueren Zählvariablen vorkommen. Ergibt sich aus den veränderten Fahrplänen anderer Agenten eine bessere Fahrplanauswahl, so kann der Agent seinen eigenen Zustand anpassen und anschließend sein Arbeitsgedächtnis an alle Nachbarn weiter senden, um ihnen die Veränderung mitzuteilen. „Eine einmal veröffentlichte Fahrplanauswahl muss von einem Verbundteilnehmer ungeachtet seiner lokalen Präferenzen zu jedem zukünftigen Zeitpunkt akzeptiert werden, wenn diese zu einer höheren globalen Güte beiträgt.“ [2]

Der Lösungskandidat bezeichnet die bis dahin beste gefundene Konfiguration, die dem Agenten bekannt ist. Bekommt er den Lösungskandidaten eines anderen Agenten zugesandt, kann er ihn mit seinem eigenen vergleichen und diesen gegebenenfalls ersetzen.

Das beschrieben Verfahren konvergiert garantiert, und besitzt die sogenannte anytime-Eigenschaft. Dies bedeutet, dass „unmittelbar nach dem Start des Verfahrens

gültige Lösungen vorliegen, welche mit zusätzlich zur Verfügung stehender Laufzeit inkrementell verbessert werden.“ [2] Unter gültigen Lösungen werden hierbei jene verstanden, die alle Constraints erfüllen. Allgemein sind Multiagentensysteme sehr robust, da kein „single point of failure“ existiert. Des Weiteren können komplexe Probleme durch die Selbstorganisation der Agenten vereinfacht werden, da Interna nicht zentral bekannt sein müssen. Außerdem kann durch Parallelisierung ein Geschwindigkeitsvorteil erreicht werden. Zugleich besteht aber eine gewisse Unsicherheit, da die Interaktionen und die Ausführung nicht vorhersehbar ist. Besonders durch die Kommunikation der Agenten kann hoher Ressourcenbedarf entstehen. Für die Einsatzplanotpimierung in virtuellen Kraftwerken sind MAS besonders interessant, da bereits eine dezentrale Struktur vorliegt.

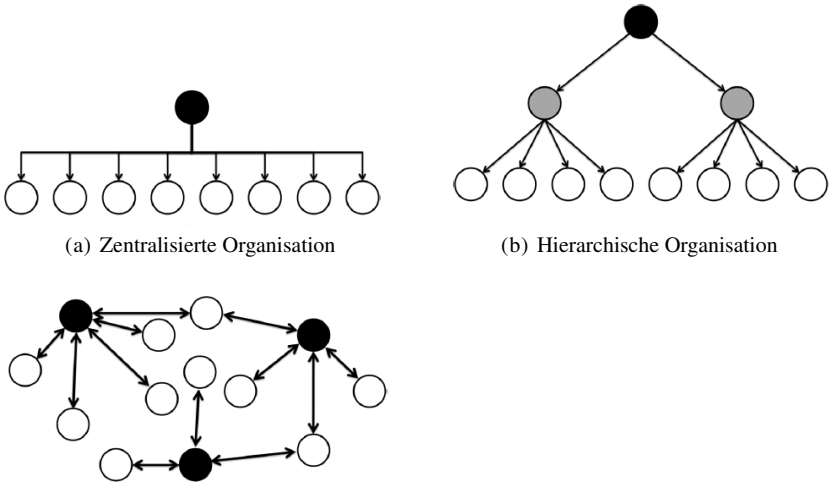
6.4.3 Organisationsformen

Organisationsformen können grundlegend in drei verschiedene Arten unterteilt werden: zentralisierte, hierarchische und dezentralisierten. Abbildung 6.9 zeigt eine schematische Darstellung der Formen.

In virtuellen Kraftwerken herrschen zentralisierte Organisationsformen vor. Sie bieten den Vorteil, dass alle relevanten Informationen, sowie der globale Systemzustand an zentraler Stelle vorliegen und beispielsweise für die Einsatzplanoptimierung genutzt werden können. Dabei wird allerdings nicht berücksichtigt, dass VKs aus zahlreichen dezentralen Kleinanlagen mitunter heterogenen Interessen bestehen. Mit steigender Komplexität der Systeme ist eine zentralisierte Lösung immer weniger geeignet. [2, Seite 4] Entscheidende Faktoren dafür sind Probleme mit Skalierbarkeit und hoher Kommunikationsaufwand zur Übermittlung von Informationen zu Kontrollzwecken. Auch die Robustheit muss beachtet werden. Zentrale Systeme besitzen betriebskritische Komponenten, die zu einem „Single Point of Failure“ werden können. Zudem müssen möglicherweise sensible Daten an eine zentrale Stelle übermittelt werden, was zu einer Verletzung der Privatsphäre führen kann. Beteiligte Akteure (z.Bsp. Anlagenbetreiber) müssen ihre Autonomie in einem zentralisierten System weitestgehend abgeben.[2, Seite 5]

Eine Alternative zur zentralen Organisation stellt die hierarchische Organisation dar. Hier ist durch eine Aufspaltung des Gesamtproblems in Teilprobleme eine parallele Verarbeitung möglich. Zudem erhöht sich im Vergleich zu zentralen Lösungen die Skalierbarkeit. Zugleich wird aber eine längere Antwortzeit benötigt und es existieren immer noch „Single Point(s) of Failure“. Ein Problem bei hierarchischen Lösungen ist zudem, dass möglicherweise suboptimale Lösungen gefunden werden.

Eine dritte Möglichkeit ist die dezentrale bzw. verteilte Organisation. Bei dieser Organisationsform existieren keine übergeordneten Instanzen mehr. Große Stärken dieser Lösungsvariante sind die einfache Skalierbarkeit und die Möglichkeit der lokalen (mehrkriteriellen) Optimierung. In einem VK würde dies beispielsweise bedeuten dass die Präferenzen der einzelnen Anlagenbesitzer besser berücksichtigt werden können, was zu mehr Fairness führt. Nachteile eines verteilten Ansatzes sind



(a) Zentralisierte Organisation

(b) Hierarchische Organisation

(c) Dezentrale Organisation

Abb. 6.9: Organisationsformen aus [4]

der große Kommunikationsbedarf, die Möglichkeit zu suboptimalen Lösungen zu kommen und ein noch vorherrschendes Akzeptanzproblem in der Energiewirtschaft. [4]

6.5 Fazit

Die Einsatzplanoptimierung in virtuellen Kraftwerken stellt ein komplexes kombinatorische Optimierungsproblem mit einer anwendungsspezifischen Zielfunktion und diversen meist technischen Einschränkungen dar. Die Lösung des Problems ist mit mathematischen Mitteln nur bedingt möglich. Daher bietet die Verwendung von heuristischen Verfahren die bessere Alternative. Aus der großen Anzahl heuristischer Methoden wurden zwei exemplarisch vorgestellt. Dabei sind evolutionäre Algorithmen eher für zentrale Kontrollstrukturen geeignet, wohingegen Multiagentensysteme sinnvoll für dezentrale Koordinationskonzepte eingesetzt werden können. Die Wahl einer geeigneten Organisationsform und dadurch bedingt eines geeigneten Algorithmus kann für das Projekt „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ erst nach der Anforderungsanalyse getroffen werden.

Literaturverzeichnis

[1] Matthias Hable. „Beitrag zur Energieeinsatzoptimierung mit evolutionären Algorithmen in lokalen Energiesystemen mit kombinierter Nutzung von Wärme- und Elektroenergie“. Dissertation. Technische Universität Dresden, 2004.

[2] Christian Hinrichs. „Selbstorganisierte Einsatzplanung dezentraler Akteure im Smart Grid“. Dissertation. Carl von Ossietzky Universität Oldenburg, 2014.

[3] Benjamin F. Hobbs. „Optimization methods for electric utility resource planning“. In: *European Journal of Operational Research* 83 (1995), S. 1–20.

[4] Sebastian Lehnhoff. *Folien zur Vorlesung "Smart Grid Management" - Carl von Ossietzky Universität Oldenburg*. 2015.

[5] Conny Neumann. *Belastetes Stromnetz: Groß-Störfall Sonnenfinsternis*. [Online; Zugriff 22.05.15]. 2015. URL: http://www.spiegel.de/wirtschaft/sonnenfinsternis-stromnetzbetreiber-tennet-praesentiert-krisenplan-a-1024061.html/.

[6] Stuart Russel und Peter Norvig. *Artificial Intelligence - A Modern Aproach*. Pearson Education, 2003.

[7] Soliman Abdel-Hady Soliman und Abdel-Aal Hassan Mantawy. *Modern Optimization Techniques with Applications in Electric Power Systems*. Hrsg. von USA Panos M. Pardalos University of Florida. Springer, 2012.

[8] P. Surekha, N. Archana und S. Sumathi. „An Integrated GA-ABC Optimization Technique to Solve Unit Commitment and Economic Dispatch Problems“. In: *Asian Journal of Scientific Research* 5 (3) (2012), S. 93–107.

[9] Martin Tröschel. „Aktive Einsatzplanung in holonischen Virtuellen Kraftwerken“. Dissertation. Carl von Ossietzky Universität Oldenburg, 2010.

[10] Ludger Winkels. „Tageseinsatzplanung dezentraler Energieerzeugungsanlagen“. Dissertation. Carl von Ossietzky Universität Oldenburg, 2009.

[11] Micheal J. Woolridge. „Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence“. In: Hrsg. von Gerhard Weiss. The MIT Press, 2004. Kap. Intelligent Agents.

Kapitel 7
Smart Grid Architecture Model

Frauke Oest

Zusammenfassung Die Entwicklung von IKT-Architekturen für Smart Grids müssen Anforderungen verschiedener Quellen beachten. Diese Quellen stellen die architektonische Sichtweise bezüglich Domänen, Funktionen und übergreifende Probleme, wie Interoperabilität und Sicherheitsfragen heraus. Aus diesem Grund wurde eine Referenzarchitektur im Kontext der European Smart Grid Mandate M/490 entwickelt, die eine Struktur mit Zones, Domains und Interoperability Layer vorschlägt. In dieser Arbeit wird das SGAM und ihre Verwendung dargelegt, sowie das empfohlene Vorgehen anhand des SGAM-Toolkits erläutert.

7.1 Einleitung

Bei der Entwicklung und dem Betrieb von Smart Grids sind verschiedene Stakeholder beteiligt. Diese haben verschiedene Sichten und Anforderungen an das System [1, S. 11]. Ein Photovoltaik-Betreiber möchte in das Netz den erzeugten Strom seiner Anlage einspeisen, wenn der Preis besonders hoch ist. Der Netzbetreiber möchte ein möglichst stabiles Netz betreiben, damit es nicht zu Stomausfällen kommt. Aber auch aus IKT-Sicht stellen die verteilten Kraftwerke eine besondere Herausforderung dar: Die verschiedenen Komponenten müssen koordiniert und so geregelt werden, dass sie das Netz für den Netzbetreiber stabil halten, also weder zu viel noch zu wenig einspeisen[3]. Die Komponenten verteilen sich auf verschiedene Domänen, die auf unterschiedliche Protokolle angesprochen werden und intern andere Standards verwenden. Das elektrische Netz soll nicht durch interne Fehler ausfallen und es muss vorallem auch gegen Angriffe durch Dritte sicher sein [1, 39ff].

Aus diesen Gründen wurde eine Referenzarchitektur entwickelt, deren Ziel es ist aus den verschiedenen Zielen der beteiligten Stakeholder in einer frühen Phase

Carl von Ossietzky Universität Oldenburg
E-mail: frauke.oest@uni-oldenburg.de

der Entwicklung eine möglichst vollständige Anforderungsanalyse und Architekturentwicklung fertigzustellen. Fehlende Standards sollen früh erkannt werden und Anforderungen sehr detailliert aufbereitet werden. Das Smart Grid Architecture Model (SGAM) wurde daher im Kontext der European Smart Grid Mandate M/490 entwickelt und schlägt ein dreidimensionales Modell bestehend aus Zonen, Domänen und Interoperabilitätsebenen vor [1, S. 11].

Diese Arbeit wird wie folgt gegliedert: Im zweiten Kapitel wird die Smart Grid Ebene mit ihren Domains und Zones erläutert, im Anschluss wird auf die Interopability Layer eingegangen. Das dritte Kapitel erläutert das Vorgehen, wie es in der empfohlenen SGAM-Toolbox vorgeschlagen wird, anhand des Metamodells der Toolbox und einem Beispiel. Das letzte Kapitel beinhaltet das Fazit.

7.2 Smart-Grid Architecture Model (SGAM)

Das Smart Grid Architecture Model ist ein dreidimensionales Modell bestehend aus der Smart Grid Ebene mit den Dimensionen Domains und Zones, die Bestandteile der fünf Interoparability Layer sind. [1, S. 12-15].

7.2.1 Smart Grid Ebene

Eine Smart Grid Ebene schließt die komplette Umwandlungskette elektrischer Energie ein und spannt sich in eine 6×5 Matrix mit Zones (dt. Zonen) und Domains (dt. Domänen) auf (siehe Abbildung 7.1).

7.2.1.1 Domains

Die Domains beschreiben den physikalischen Aspekt des Stromnetzes. Bestandteile der Domain sind Generation (dt. Generierung), Transmission (dt. Übertragung), Distribution (dt. Verteilung), DER (distributed electrical resources, dt. verteilte elektrische Ressourcen) und Customer Premises (dt. Privatgrundstücke). Konkret bedeutet dies:

Generation repräsentiert das Erzeugen von Energie in großen Mengen, die typischerweise mit dem Übertragungssystem verbunden sind. Die Energieerzeugung kann durch fossile, nukleare, und regenerative Ressourcen erfolgen.

Transmission repräsentiert die Infrastruktur des Energietransports über lange Distanz.

Distribution repräsentiert die Infrastruktur, die für die Verteilung an den Endverbraucher zuständig ist.

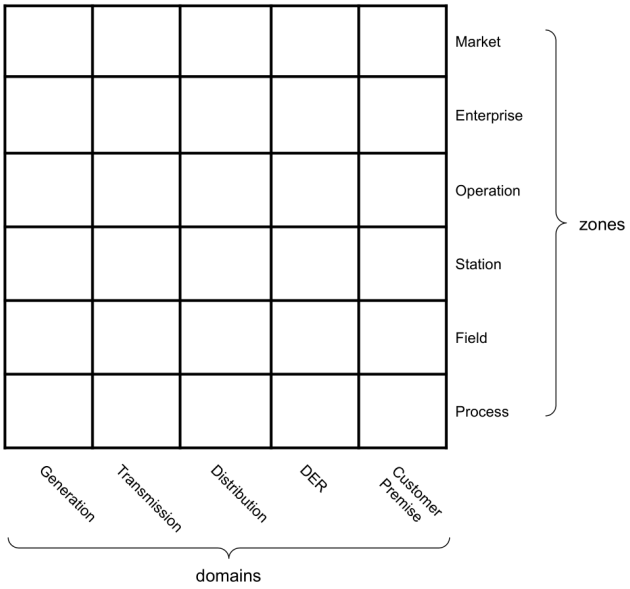


Abb. 7.1: SGAM Domains und Zones [5, S. 26]

DER repräsentieren kleine verteilte elektrische Ressourcen (3-10000kW), die direkt ans öffentliche Netz angeschlossen sind. Sie können sowohl Erzeuger, als auch Verbraucher sein. Ein Erzeuger ist beispielsweise eine Windkraftanlage und sie wird durch Systeme bzw. Unternehmen wie Übertragungsnetzbetreiber (TSO) oder Verteilnetzbetreiber (DSO) gesteuert.

Customer Premises beinhalten sowohl die Elektrizität des Endbenutzers, als auch die Elektrizität eines lokalen Erzeugers. Die Grundstücke können industriell, kommerziell oder privat genutzt werden. Die Ezeuger von Elektrizität sind kleiner als im DER. Typischerweise sind das Photovoltaik-Anlagen, Batterien, Mikroturbinen oder die Speicherung von Energie in elektrischen Fahrzeugen.

Anzumerken sei, dass DER und Customer Premises zwar beide die Erzeugung/der Verbrauch (Speicher) von Elektrizität beinhalten, aber doch getrennte Domains sind. Der Unterschied ist, ddass eine vollständige Einspeisung das Ziel von Erzeugern/Verbrauchern im DER ist. Im Customer Premises erzeugen/verbrauchen die Anlagen hauptsächlich lokal und nur bei Über-/Unterproduktion wird ins Netz eingespeist [1, S. 12-15].

7.2.1.2 Zones

Im SGAM repräsentieren die Zones die hierarchischen Ebenen des Energieversorgungsnetzmanagements.

Process beinhaltet die physikalische und chemische Umwandlung von Energie, wie beispielsweise von Wind- zu elektrischer Energie. Außerdem beinhaltet es das direkt von der Umwandlung betroffene physische Equipment, wie Generatoren, Kabel, elektrische Verbraucher, Sicherungen, sowie Sensoren.

Field beinhaltet das Equipment, das den Prozess der Stromversorgung schützt, steuert und überwacht. Beispiele für solches Equipment sind Schutzrelais oder intelligente, elektronische Geräte, die Prozessdaten der Stromversorgung aufnehmen und bentutzen.

Station repräsentiert die Flächenaggregration auf Field Ebene, um beispielsweise Daten lokaler SCADA Systeme für Umspannwerksautomation zu konzentrieren oder funktional zu aggregieren.

Operation nehmen Stromversorgungssteurungsoperationen in der entsprechenden Domain auf. Typische Systeme, die sich auf dem Operation Layer befinden sind beispielsweise das Energiemanagement System (EMS), das Vertriebmanagement System (DMS), sowie das System zur Steuerung des virtuellen Kraftwerks.

Enterprise besteht aus kommerziellen und organisatorischen Prozessen, Dienstleistungen und Infrastrukturen für Unternehmen im Energiebereich, wie Logistik, Personalmanagement, Rechnungen, etc.

Market reflektiert die Markttätigkeit, die während der Energieumwandlungskette stattfinden kann, wie Energiehandel oder Einzelhandel [1, S. 12-15].

Die Unterteilung der Zones basiert auf Aggregation und funktionaler Aufteilung. Die Aggregation besteht aus multiplen Aspekten: Zum einen gibt es die Datenaggregation, bei dem die Daten aus der Field Zone in der Station Zone aggregiert oder konzentriert werden. Dadurch wird die Menge der Daten, die zur Operation Zone übertragen werden muss, verringert.

Zum anderen gibt es die räumliche Aggregation. Hierbei werden kleine Orte zu einem großen Gebiet zusammengefasst. Ein Beispiel für eine solche Aggregation ist die Zusammenfassung von mehreren DER zu einem Kraftwerk.

7.2.2 Interoperabilität

Das SGAM besteht aus fünf dieser Ebenen. Die Ebenen werden in Abbildung 7.2 dargestellt und repräsentiert Geschäftsziele, Prozesse, Funktionen, Informationsaustausch, Kommunikationsprotokolle und Komponenten.

Business repräsentiert die Geschäftssicht auf den Informationsaustausch im Zusammenhang von Smart Grids. SGAM kann dazu genutzt werden die regulatorischen und ökonomischen Strukturen abzubilden. Involviert sind Geschäftsmodel-

le, Anwendungsfälle, Geschäftsportfolios, Strategien, Marktparteien, Geschäftsfähigkeiten und -prozesse.

Function beschreibt die System-Anwendungsfälle, -Funktionen und ihre Beziehungen aus dem architektonischen Standpunkt. Die Funktionen werden unabhängig vom Akteur oder physikalischen Implementierungen in Anwendungen, Systemen und Komponenten repräsentiert. Sie werden abgeleitet in dem die vom Akteur unabhängigen Anwendungsfallfunktionalitäten extrahiert werden.

Information beschreibt die Informationen, die zwischen Funktionen, Services und Komponenten benutzt und ausgetauscht werden. Es enthält Informationsobjekte und die tieferliegenden kanonischen Datenmodelle. Diese repräsentieren die allgemeine Semantik für Funktionen und Dienste, die den interoperablen Informationsaustausch über Kommunikationsmittel erlauben.

Communication Der Schwerpunkt des Communication Layers ist die Bechreibung von Protokollen und Mechanismen zum interoperablen Informationsaustausch zwischen Komponenen im Hinblick auf den tieferliegenden Anwendungsfall, Funktion oder Dienst und im Hinblick auf die verwandten Informationsobjekte oder Datenmodelle.

Component Der Schwerpunkt des Compontent Layers liegt auf der physikalischen Verteilung aller teilnehmenden Komponenten im Smart Grid Kontext. Dies schließt System- und Geräte-Akteure, Stromsystem Equipment, Schutz und Fernsteuerungen, Netzwerkinfrastruktur und jegliche Art von Computer ein [1, S. 12-15].

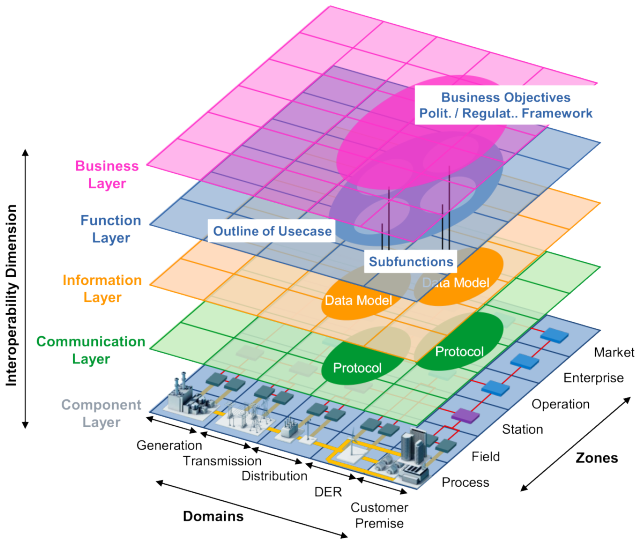


Abb. 7.2: SGAM Domains und Zones [5, S. 24]

7.3 SGAM Toolbox

Für die Modellierung des Smart Grid Architecture Models reicht es prinzipiell aus Microsoft Powerpoint oder Microsoft Visio zu verwenden und die Komponenten dort darzustellen. Allerdings kann dies zu ungewollten Interpretationen führen und damit zu einer inkonsistenten Verwendung dieses Referenzmodells. Um dies zu verhindern, ist es daher ratsam die SGAM Toolbox zu verwenden. Sie ist ein PlugIn für Sparx Enterprise Architect. Diese Software unterstützt modellgetriebende Generierung (MDG), wodurch eine Struktur vorgegeben wird mit der eine konstistente Verwendung des SGAM möglich ist [2].

Die Strukur der SGAM-Toolbox basiert auf dem Metamodell aus Abbildung 7.3. In ihr werden die Elemente des SGAM und deren Relationen abgebildet.

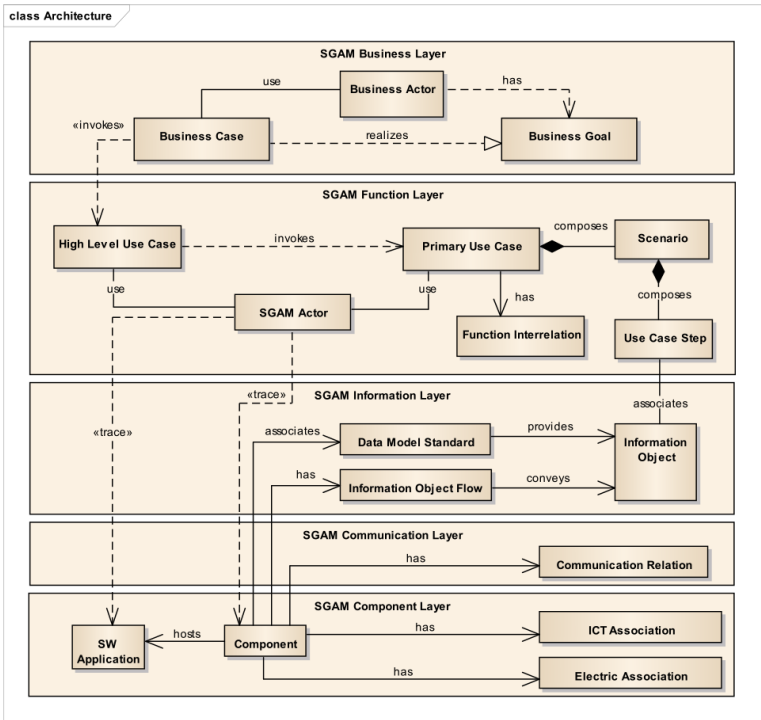


Abb. 7.3: SGAM Metamodell [4, S. 9]

Im Folgenden wird das Metamodell anhand eines Beispiels erläutert, wobei die Energiedomäne vernachlässigt werden soll. Die Ausgangssituation besteht aus einer Anlage, die aus Blockheizkraftwerk, Wärmepumpe und Wärmespeicher besteht und dementsprechend Wärme und elektrische Energie produzieren kann.

7.3.1 Business Layer

Der Business Layer besteht im Wesentlichen aus Business Actors (Geschäftsakteure), Business Goals (Geschäftsziele) und Business Use Cases. Business Goals sind allgemeine Anforderungen, aus denen dann Business Use Cases (Geschäftsanwendungsfall) abgeleitet werden. Business Use Cases beschreiben den Geschäftsprozess, der von Actors ausgeführt wird. Im Business Layer existiert keine technische Sicht auf das System.

In diesem Beispiel in Abbildung 7.4 existieren zwei Actors: der facility operator (Anlagenbetreiber) und der DSO (Verteilnetzbetreiber). Diese können in Beziehung zu einander stehen, wie in diesem Fall durch einen Vertrag. Hier haben beide verschiedene Business Goals. Der Facility Operator hat als Ziel seine Energiekosten zu optimieren. Dies erreicht er indem er flexible Verbraucher bereitstellt, was zugleich der dazugehörige Business Use Case ist [4, S. 14-22].

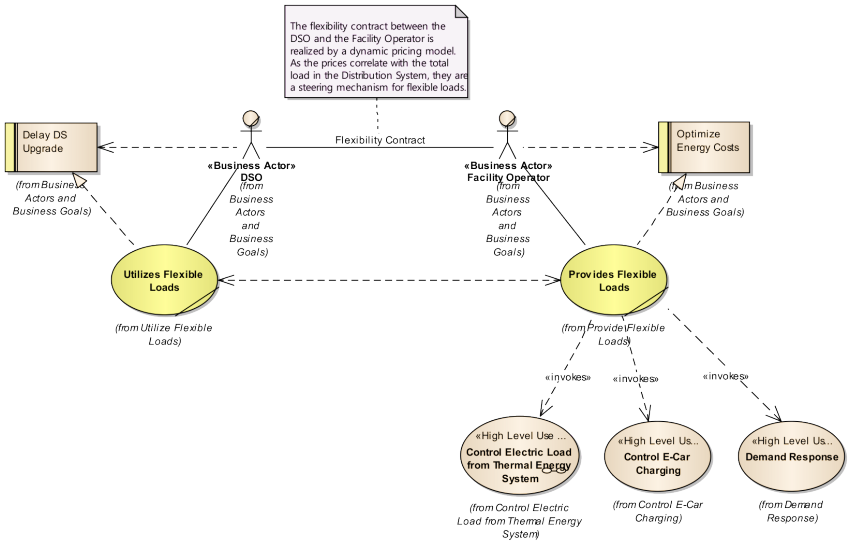


Abb. 7.4: Business Use Case Model [4, S. 15]

Im nächsten Schritt müssen die High Level Use Cases (HLUC) aus den Business Use Cases ermittelt werden. Ein HLUC beschreibt die allgemeine Idee einer Funktion mit ihren Actors. In der SGAM Umgebung sind die verschiedenen HLUC Teilaufgaben, die zur Erfüllung des Business Use Cases benötigt werden. So kann durch die Steuerung vom elektrischen Verbrauch des WES ein flexibler Verbrauch ermöglicht werden.

Die bisher durchgeführte Use Case Analyse reicht jedoch nicht aus, um die exakte Lage in der Smart Grid Ebene zu bestimmen. Die genaue Lage benötigt die Modellierungen aus dem Function Layer, sodass der HLUC im Business Layer alle Bereiche abdeckt, die die Elemente im Function Layer einnehmen. Der Business Use Case muss dann alle Bereiche der Smart Grid Ebene seiner HLUC abdecken. Der Business Use Case in Abbildung 7.5 deckt die Domains Distribution bis Customer Premise und die Zones Operation bis Process ab, weil unter anderem Der HLUC „Control Electric Load from Thermal Energy System“ die Bereiche Distribution, DER, Operation, Station, Field und Process abdeckt. [4, S. 14-22].

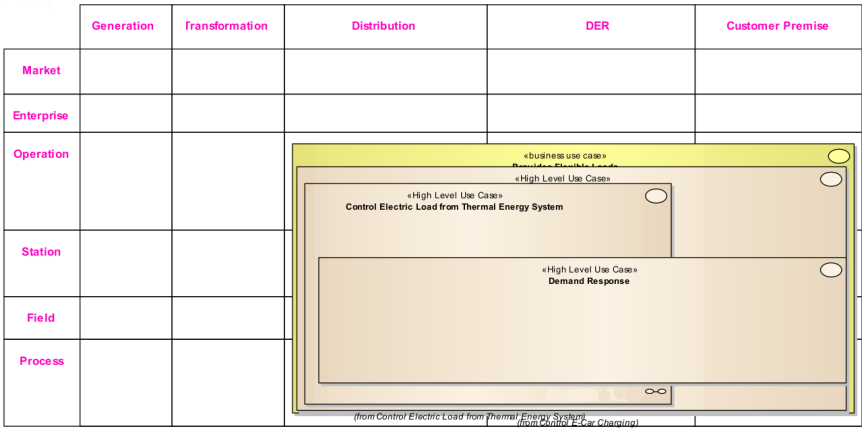


Abb. 7.5: Business Layer nach Use Case Analyse [4, S. 22]

7.3.2 Function Layer

Für den Function Layer werden die ermittelten High Level Use Cases in weitere Teilaufgaben zu Primary Use Cases (PUC) unterteilt. Hier wird dazu der HLUC „Controll Electric Load from Thermal Energy System“ in weitere Aufgaben unterteilt. Für die Steuerung vom Verbrauch des Wärme-Energie-Systems (HLUC: „Controll Electric Load from Thermal Energy System“) müssen Daten aufgenommen (PUC: „Data Acquisition“), ein Operationsplan erstellt (PUC: „Operation Plan“), sowie die Wärmeerzeugung, die Wärmepumpe und das Blockheizkraftwerk, reguliert werden (PUC: „Controll Heatpump“ und „Control CHP“). Im nächsten Schritt werden die Primary Use Cases in Beziehung zu einander gesetzt und ihre jeweiligen Actors ermittelt. Dies reicht bereits aus, um den Function Layer zu erstellen.

Im Function Layer werden die Primary Use Cases zerlegt, in Relation zueinander gesetzt und die dazugehörigen SGAM Actors ermittelt. Auch hier kann die Posi-

tion der Elemente auf dem Function Layer in Abbildung 7.6 noch nicht endgültig festgelegt werden, da die Position der Actors im Component Layer bestimmt wird.

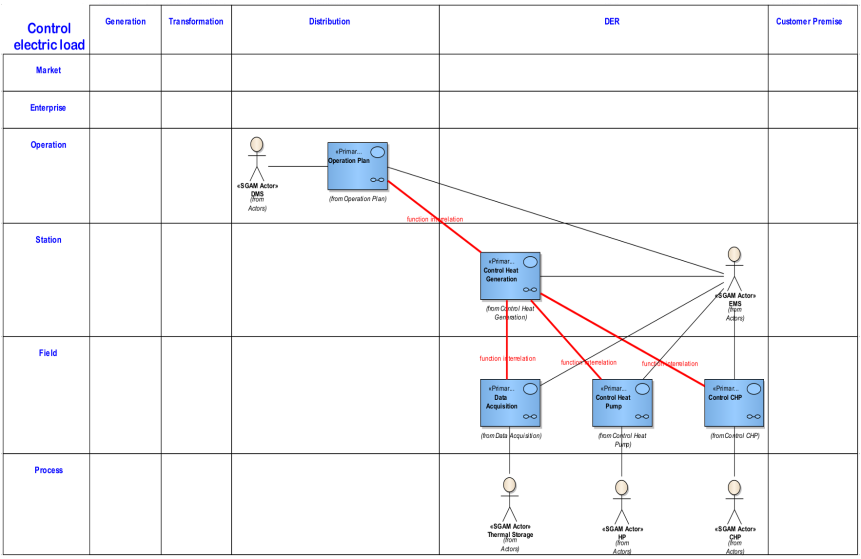


Abb. 7.6: Function Layer [4, S. 21]

Für das weitere Vorgehen müssen die Primary Use Cases detaillierter beschrieben werden. Dies geschieht über Szenarien des Primary Use Cases. Diese Szenarien bestehen wiederum aus Use Case Steps, mit denen der Ablauf innerhalb des Primary Use Cases beschrieben wird. Daher wird für die Benutzung der SGAM Toolbox vorgeschlagen ein Szenario sowohl als Sequenzdiagramm, als auch als Aktivitätsdiagramm zu modellieren.

Für den Primary Use Case „operation plan“, an dem das distributed management system (DMS)und energy management system (EMS) beteiligt sind ist das Beispielszenario folgendermaßen: Der EMS meldet sich beim DMS an, er erlaubt dann die Verbindung. Als Informationsobjekt werden hierbei Anmeldedaten versendet. Im späteren Verlauf fordert der EMS eine Energiepreistabelle vom DMS an. Das Informationsobjekt ist dementsprechend die Energiepreistabelle, die übertragen werden muss [4, S. 14-22].

7.3.2.1 Component Layer

Für die Entwicklung in der SGAM Toolbox wird in [4] vorgeschlagen nach der Modellierung des Function Layers mit dem Component Layer fortzufahren. Er bildet die Stuktur für den Information Layer und dem Communication Layer.

Zur Entwicklung des Component Layers werden die SGAM Actors, die in den Primary Use Cases im Function Layer entwickelt wurden, verwendet. Die SGAM Actors werden durch ihre phyikalischen Komponenten, sowie ihrer Steuerungseinheit abgebildet. Wie in Abbildung 7.7 bestehen CHP (Blochheizkraftwerk), DMS (Vertriebsmanagement-System), HP (Wärmepumpe) und Thermal Storage aus phyisikalischen Komponenten und Steuerungseinheiten. Allerdings zeigt die Abbildung 7.7 auch, dass der EMS beispielsweise eine reine Steuerungseinheit ist. Die Kom-

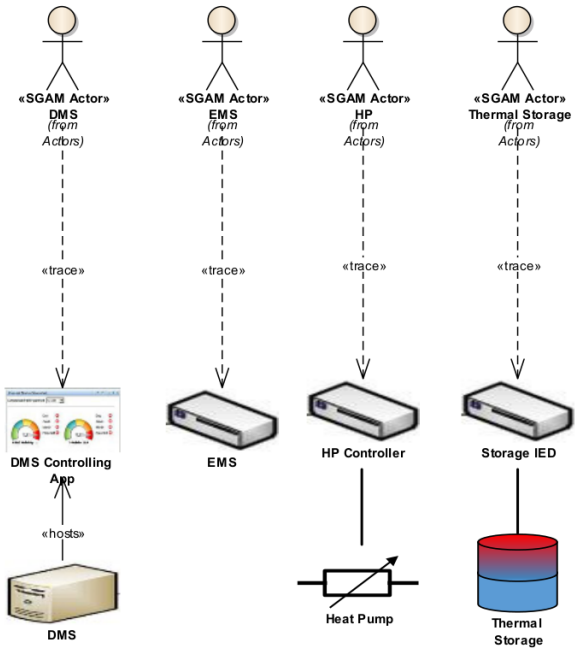


Abb. 7.7: Abbilden von Actors auf Komponenten [4, S. 24]

ponenten werden auf dem Component Layer in den entsprechenden Zones und Domains abgebildet und gemäß der Primary Use Cases miteinander verbunden. Sollten zwei Komponente, wie EMS und DMS in Abbildung 7.8 über eine Remote-Verbindung kommunizieren, wird dies durch eine Wolke dargestellt. Wenn Komponenten am Stromnetz hängen, wird dies durch ein Rechteck dargestellt und die Spannung angegeben[4, S. 23-25].

7.3.2.2 Communication Layer

Beim Communication Layer wird das Gerüst des Component Layers zu einem großen Teil übernommen. Allerdings beschränkt es sich hier nur auf die Steuerungs-

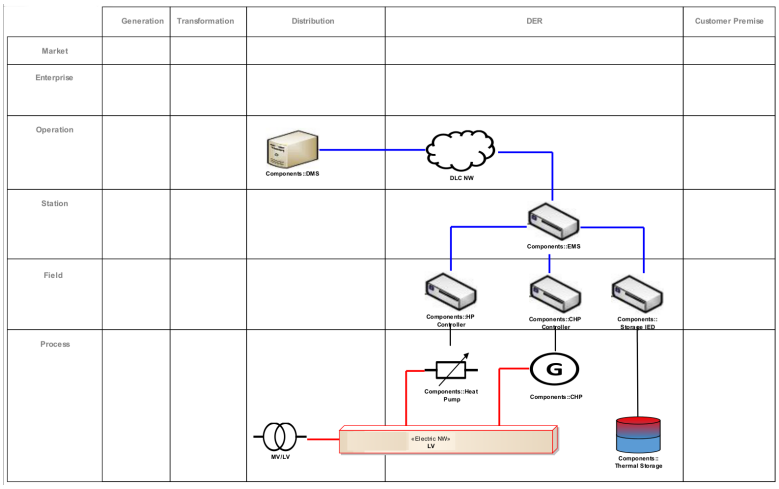


Abb. 7.8: Abbildung auf SGAM Ebene [4, S. 25]

komponenten, die aus IKT-Sicht angesprochen werden können, weshalb nicht der Wärmespeicher direkt angesprochen wird, sondern die Steuerung des Wärmespeichers. In diesem Layer wird bestimmt, welche Technologie und welches Protokoll zur Kommunikation zwischen den Komponenten verwendet werden.

Dem Beispiel folgend werden in Abbildung 7.9 exemplarisch als Technologie das Ethernet oder ADSL, als Protokoll Webservices und OPC-UA genannt [4].

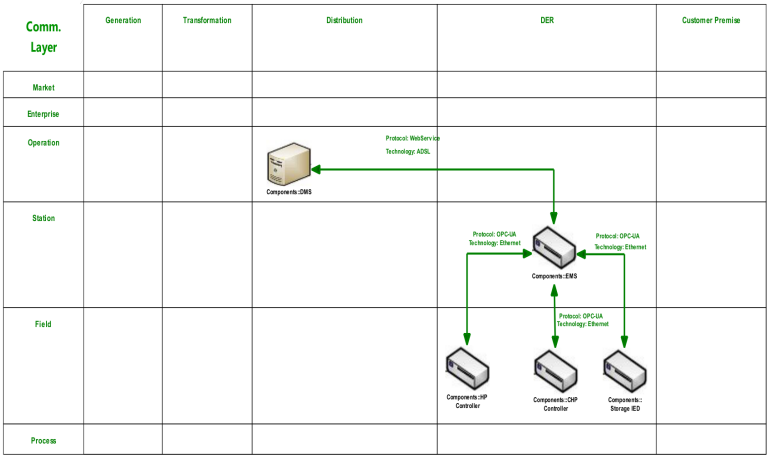


Abb. 7.9: Communication Layer [4, S. 29]

7.3.2.3 Information Layer

Die Entwicklung des Information Layer erfolgt in drei Schritten. Im ersten Schritt wird der Business Context ermittelt. Er dient dazu den Informationsfluss der Informationsobjekte aufzuzeigen. Diese Informationsobjekte werden, wie in Abbildung 7.10, an die Verbindungen zwischen den Komponenten geschrieben.

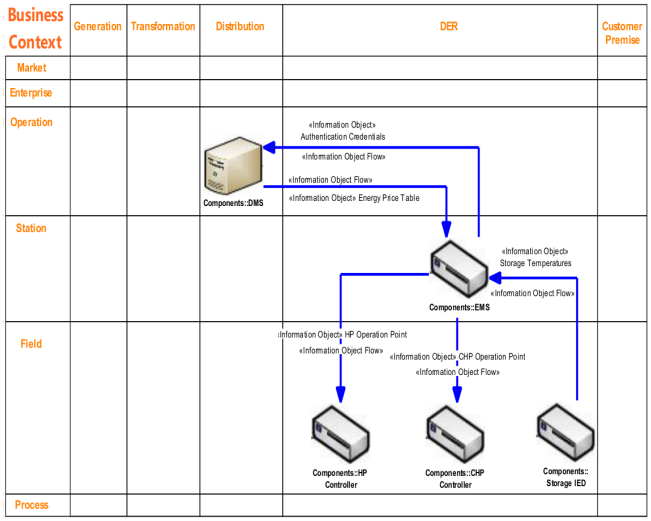


Abb. 7.10: Business Context des Information Layer [4, S. 26]

Im zweiten Schritt wird versucht für die Informationsobjekte gemeinsame Standards bzw. Datenmodelle zu finden. In diesem Beispiel stellt der „Energy Price Standard“ das Datenmodell für die „Energy Price Table“ und „Authentication Credentials“ dar.

Im letzten Schritt werden die ermittelten Datenmodelle in einem seperaten Information Layer gezeichnet, wie in Abbildung 7.11 dargestellt, welches die Elemente des Business Context überdeckt [4, S. 25-28].

7.4 Fazit

Das Smart Grid Architecture Model ist eine Möglichkeit die komplexen Zusammenhänge in einem Smart Grid übersichtlich darzustellen und wird in der fühen Entwicklungsphase eingesetzt. Durch die Aufteilung in Zones und Domains wird schnell deutlich, wenn Komponenten, die in Verbindung stehen, verschiedene Standards verwenden und es kann entsprechend früh reagiert werden. Durch die verschiedenen

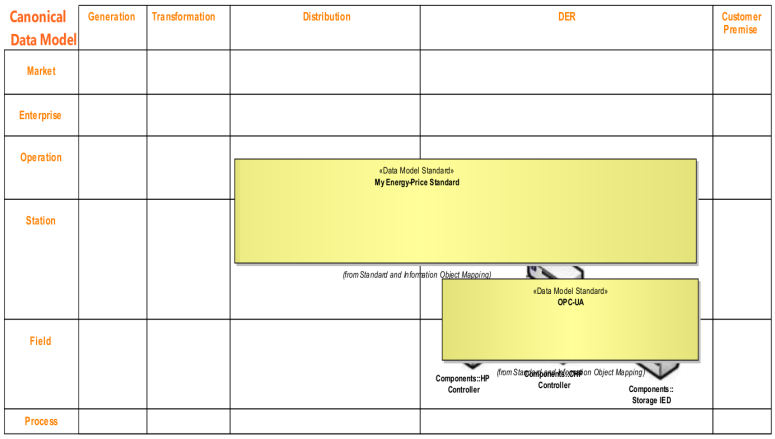


Abb. 7.11: Canonical Data Model View [4, S. 28]

Interoperabilitätsebenen ist es auch für Stakeholder ohne großes IKT-Wissen möglich an der Anforderungserhebung auf Ebene des Business- und Function Layers mitzuwirken.

Für die Projektgruppe „Dynamische Portfoliooptimierung eines virtuellen Kraftwerks“ ist das Smart Grid Architecture Modell somit gutes Vorgehen, um ganzheitliches Requirements Engineering zu betreiben. Zusätzlich wird empfohlen, langfristig SGAM mit Hilfe der SGAM Toolbox umzusetzen, da diese eine gute Hilfestellung ist und damit die konstistente Benutzung des SGAM gewährleistet wird.

Literaturverzeichnis

[1] SG-CG. *SGAM User Manual - Applying, testing & refining the Smart Grid Architecture Model*. Smart Grid Coordination Group. Nov. 2014.

[2] Christian Dänekas u. a. *Durchgängige Werkzeugunterstützung für das EU Mandat M/490: Vom Anwendungsfall bis zur Visualisierung*. OFFIS - Institut für Informatik, Zentrum für Anwendungsorientierte Smart Grid Privacy, Sicherheit und Steuerung, Okt. 2014.

[3] Sebastian Lehnhoff. „Smart Grid Management: Aufbau und Betrieb elektrischer Energieversorgung“. Apr. 2015.

[4] Christian Neureiter. *Introduction to the SGAM Toolbox*. Apr. 2014.

[5] Mathias Uslar u. a. *Standardization in Smart Grids - Introduction to IT-related Methodologies, Architectures and Standards*. Juli 2013.

Kapitel 8
Common Information Model

Stephan Balduin

Zusammenfassung Für die Umsetzung der Projektziele im Rahmen der Projektgruppe müssen geeignete Modellierungswerkzeuge gefunden werden. Das Smart Grid Architecture Model ist eine gute Wahl für alle Systeme, mit Energieversorgung zu tun haben. Konkreter Informations- und Kommunikationsstandard werden allerdings nicht vorgegeben. Das Common Information Model ist ein Informationsstandard, der mit einer Erweiterung auch als Kommunikationsstandard verwendet werden kann. In diesem Beitrag soll das Common Information Model näher beschrieben und der Nutzen für die Projektgruppe diskutiert werden.

8.1 Einleitung

Im Kontext vom Energiesystemen wird aktuell viel über das Thema Smart Grid diskutiert [5]. Ein wichtiges Modellierungswerkzeug für Smart Grids ist das *Smart Grid Architecture Model* (SGAM). Zwei Teilaspekte von SGAM sind das *Information Layer* und das *Communication Layer*, in denen einmal die Darstellung und einmal der Austausch von Informationen und Daten festgelegt wird. Damit Systeme wie das Smart Grid Architecture Model auch zukünftigen Anforderungen gewachsen sind, benötigen sie Standards. Ein wichtiger, wenn nicht der Standard von SGAM ist das IEC 61970/61968 *Common Information Model* (CIM). Dieses bietet ein mächtiges Datenmodell und ist in der Lage, verschiedene Schnittstellenspezifikationen und -technologien abzubilden. Es kann somit sowohl im *Information Layer*, als auch im *Communication Layer* zum Einsatz kommen. Im Rahmen der Projektgruppe soll eine Software entwickelt werden, die Bestandteil eines Smart Grid sein könnte. In diesem Beitrag soll zunächst das Thema Standards im Allgemeinen behandelt und anschließend das Common Information Model vorgestellt werden. Zum Schluss

Carl von Ossietzky Universität Oldenburg
E-mail: stephan.balduin@uni-oldenburg.de

gibt es ein Fazit, in dem erläutert wird, ob und wenn ja, welche Komponenten des Common Information Model in der Projektgruppe umgesetzt werden könnten.

8.2 Standards

Speziell im Kontext von Smart Grids, aber auch im Allgemeinen, sind Standards unabdingbar. Ohne Standardisierungen würden extrem hohe Kosten entstehen, wenn neue Komponenten oder Anwendungen in ein bestehendes System integriert werden sollen, da es unzählige Schnittstellen zu berücksichtigen gäbe [5]. In diesem Abschnitt werden Informations- und Kommunikationsstandards gegenüber gestellt und voneinander abgegrenzt.

8.2.1 Informationsstandards

Informationsstandards definieren die Art und Weise, wie Informationen dargestellt werden. Dies lässt sich am besten mit Informationsmodellen beschreiben [4]. Ein Informationsmodell repräsentiert Konzepte, Beziehungen, Operationen, Einschränkungen und Regeln in einer festgelegten Form, sodass der Leser in der Lage ist, die Semantik des dargestellten Systems zu verstehen. Informationsmodelle werden mit einer Modellierungssprache beschrieben. Dabei handelt es sich um eine formale Syntax, in der festgelegt ist, wie bestimmte Semantiken ausgedrückt werden können. Beispiele für solche Modellierungssprachen sind die grafische Notation des *Entity-Relationship* (ER-) Modells und die *Unified Modeling Language* (UML). Das ER-Modell besteht aus einer Reihe von grafischen Bausteinen, die speziell dafür geeignet sind, relationale Daten(bank)schemata darzustellen. Allgemeiner als das ER-Modell ist UML, eine universale Modellierungssprache, die eine Reihe von Diagrammen definiert, mit denen sich Systeme beschreiben lassen. Eine Übersicht dieser Diagramme ist in Abbildung 8.1 zu sehen. UML unterscheidet zwei Arten von Diagrammen. Es gibt Diagramme, die die Struktur beschreiben und Diagramme, die das Verhalten eines Systems beschreiben. Aufgrund seiner Vielseitigkeit wurde UML zunächst von der *Object Management Group* (OMG) und später auch von der *International Organisation for Standardization* (ISO) als Standard übernommen.

8.2.2 Kommunikationsstandards

Kommunikationsstandards definieren Protokolle für die Kommunikation zwischen zwei oder mehreren Komponenten. Eine Komponente kann sowohl ein gewöhnlicher Rechner sein, als auch ein anderes kommunikationsfähiges Endgerät. Ein Kommunikationsprotokoll beschreibt den Ablauf der Kommunikation und beinhaltet Syntax,

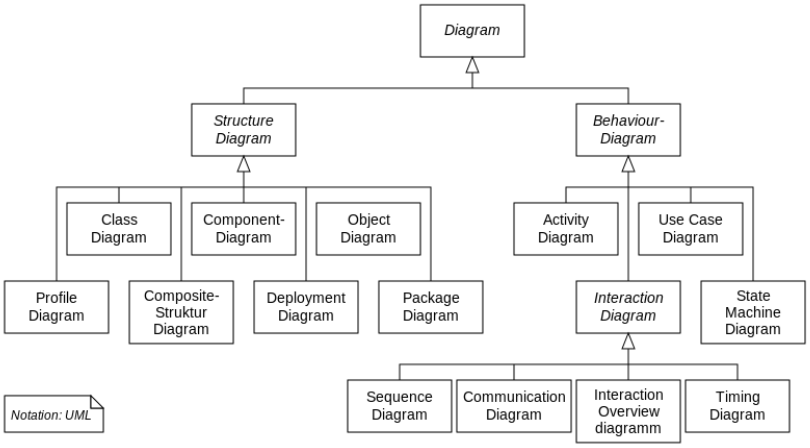


Abb. 8.1: UML-Diagramme (Quelle: de.wikipedia.org/wiki/Unified_Modeling_Language)

Semantik und Synchronisation in Form von Regeln. Die Umsetzung von Protokollen kann auf verschiedenen Ebenen stattfinden. Einerseits müssen Komponenten Hardware-seitig kommunizieren können, andererseits ist auf Software-Ebene ebenfalls eine Vereinbarung des Nachrichtenformats notwendig. Bekannte Beispiele für Kommunikationsprotokolle sind das *Internet Protocol* (IP) und das *Transmission Control Protocol* (TCP) für die Kommunikation im Internet. Ein weiteres Kommunikationsprotokoll, welches den Austausch von Daten zwischen zwei Anwendungen ermöglicht, ist das *Extensible Markup Language Protocol* (XMLP). Dieses verwendet XML als Datenformat. Mit XML lassen sich strukturierte Daten textuell beschreiben. Das dabei entstandene Dokument lässt ist in der Regel sowohl von Menschen, als auch von Maschinen lesbar. XML-Dokumente bestehen aus Zeichenketten, die entweder *Markup* oder *Content* darstellen. Markup wird meist durch eckige Klammern (< und >) gekennzeichnet. Mittels Markup lässt sich der Content des Dokuments strukturieren.

8.3 Common Information Model

Nachdem nun Informations- und Kommunikationsstandards gegenüber gestellt wurden, soll im Folgenden das Common Information Model näher beschrieben werden. Das Common Information Model ist ein von der *Distributed Management Task Force* spezifizierter offener Standard. Offen bedeutet, dass der Standard von der Öffentlichkeit geprüft und verwendet werden kann, er keine Abhängigkeiten von Komponenten

besitzt, die nicht offen sind und vollständige Implementierungen kostenfrei verfügbar sind [3]. CIM soll die Verwaltung verteilter heterogener Systeme unterstützen.

8.3.1 Organisation

Die *Distributed Management Task Force*[1] (DMTF) ist eine Organisation, die IT-Industriestandards entwickelt. Die DMTF wurde im Jahr 1992 gegründet und setzt sich aus über 200 führenden Industrieorganisationen aus mehr als 40 verschiedenen Ländern zusammen. Darunter befinden sich zum Beispiel Advanced Micro Devices, Oracle, IBM, Intel und Microsoft. Außerdem arbeitet die DMTF auch mit anderen Normierungsinstitutionen zusammen. Neben dem Common Information Model ist die Distributed Management Task Force auch für die Veröffentlichungen vieler weiterer Standards verantwortlich, zum Beispiel Cloud Management, Common Diagnostic Model, Open Virtualization Format und Web Based Enterprise Management.

8.3.2 CIM Standards

Das Common Information Model wird innerhalb drei verschiedener IEC Standardreihen spezifiziert [5]. Diese Standards konzentrieren sich auf folgende Themengebiete:

- Der Standard **IEC 61970** beschäftigt sich mit *Energy Management System Application Program Interfaces* und bietet unter anderem ein umfassendes Datenmodell und ein Integrationsframework.
- Der Standard **IEC 61968** trägt den Namen *Application Integration at Electric Utilities - System Interfaces for Distribution Management* und ist auf die Verbindung zweier Systeme und den Austausch von (XML-) Nachrichten spezialisiert.
- Der Standard **IEC 62325** befasst sich mit dem Gebiet *Framework for Energy Market Communications*. Dies beinhaltet die Kommunikation zwischen *Market Participants* und *Market Operator*, sowie zwischen verschiedenen *Market Operators*, beispielsweise dem europäischen und dem amerikanischen Markt.

In Abbildung 8.2 sind die Standards aufgeführt, die zu diesen Standardreihen dazu gehören.

8.3.3 CIM Architektur

Beim Common Information Model handelt es sich um eine objektorientierte Architektur [2]. Dies bedeutet, dass Objekte in der realen Welt als Instanzen dargestellt werden. Eine Klasse repräsentiert eine bestimmte Kategorie von Objekten. Objekte haben bestimmte Eigenschaften, die durch Attribute dargestellt werden und können

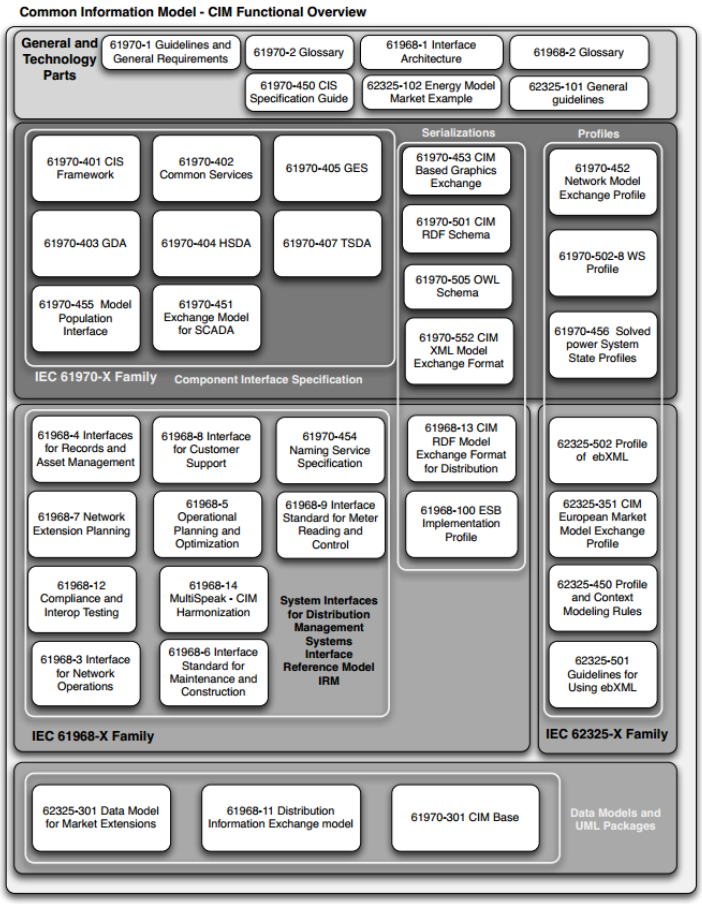


Abb. 8.2: Übersicht der einzelnen CIM Teilstandards von IEC 61968, 61970 und 62325. Stand 2011 [5]

bestimmte Aktionen ausführen, die durch Methoden repräsentiert werden. Außerdem können Objekte in Beziehung zu anderen Objekten stehen. So könnte ein Objekt zum Beispiel eine Spezialisierung (*is-a*-Beziehung) oder ein Teil (*is-part-of*-Beziehung) eines anderen Objektes sein.

8.4 CIM Spezifikation

In der CIM Spezifikation wird beschrieben, wie Dinge innerhalb von CIM dargestellt werden. Dazu definiert die CIM Spezifikation Syntax und Regeln, ein Meta-Schema,

eine Syntax-Sprache und Namensmechanismen. Die CIM Spezifikation definiert außerdem jedes Element und dessen Regeln innerhalb des Meta-Schemas. In diesem Abschnitt werden das Meta-Schema und die Syntax-Sprache *Managed Object Format* (MOF) näher beschrieben.

8.4.1 Meta-Schema

Das CIM-Meta-Schema ist in Abbildung 8.3 zu sehen. Das Meta-Schema beschreibt, wie Dinge in CIM dargestellt werden sollen. Als Wurzelelement ist im Meta-Schema das *Named Element* aufgeführt, das ein Attribut *Name* besitzt. Von diesem sind die (Haupt-)Elemente *Schema*, *Class*, *Property* und *Method* abgeleitet. Ein *Schema* dient zu Administrationszwecken und definiert die Namensgebung für Klassen. Klassen müssen einen einzigartigen Namen innerhalb ihres Schemas besitzen. Eine *Class* ist der Prototyp für eine bestimmte Art von Objekten. Sie beschreibt Eigenschaften und Verhalten dieser Objekte in Form von *Properties* und *Methods*. Eine *Property* ist ein Wert, der eine bestimmte Eigenschaft einer Klasse beschreibt und besitzt einen Namen, einen Datentyp, einen Wert und optional einen Standardwert. Datentypen einer *Property* sind gängige Datentypen, wie z. B. Integer, Boolean oder String. Eine *Method* hingegen beschreibt eine Operation, die von der Klasse ausgeführt werden kann. *Methods* besitzen einen Namen und einen Rückgabewert, sowie optionale Eingabe- und Ausgabeparameter. Eine besondere *Property* ist die *Reference*. Hierbei handelt es sich um einen Zeiger, der auf eine andere Instanz einer Klasse verweist. Die *Association* hingegen ist eine Klasse, die zwei oder mehr *References* enthält und dadurch das Verhältnis dieser Klassen zueinander repräsentiert. Die *Reference* ist in diesem Kontext dafür zuständig, die jeweilige Rolle beteiligter Klassen zu beschreiben. Zur Eventbehandlung innerhalb von CIM gibt es die *Indication*-Klasse. Diese kann ebenfalls *Properties* und *Methods* besitzen und repräsentiert das Auftreten eines Events. [2]

8.4.2 CIM Managed Object Format

Das *Managed Object Format* ist eine Syntax-Sprache, die auf der *Interface Definition Language* (IDL) basiert und eine textuelle Beschreibung von CIM Objekten ermöglicht. Insbesondere lassen sich *Classes*, *Associations*, *Properties*, *References*, *Methods* und *Instances* beschreiben. Zusätzlich ist es gestattet, Kommentare hinzuzufügen. In Abbildung 8.4 ist ein Beispiel für die textuelle Beschreibung einer Klasse mittels MOF dargestellt. Vor der Entwicklung der cimXML Spezifikation wurde das MOF außerdem als Kommunikationsformat für die Darstellung von CIM Objekten verwendet. [2]

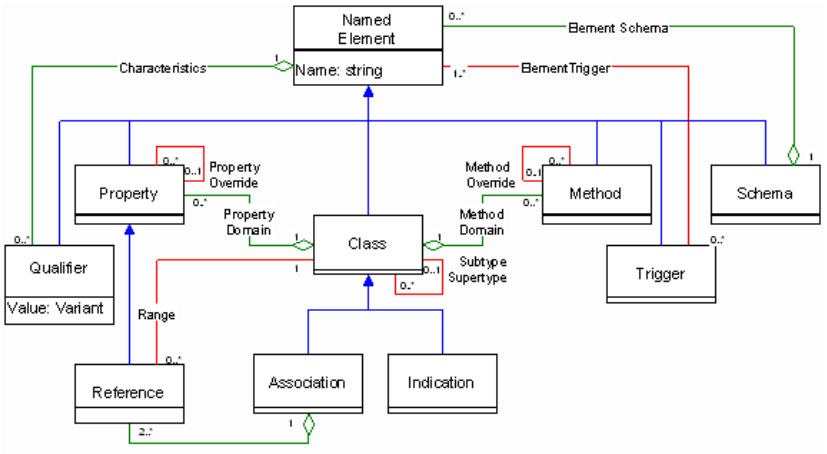


Abb. 8.3: CIM Meta-Schema

```
[Version ("2.7.0"), Experimental, Description (
 "A CIM is a type of CIM_WBEMService "
 "that instruments one or more aspects of the CIM Schema. "
 "A CIM_Provider operates at the request of the"
 "CIM_ObjectManager to perform operations on CIM objects. "
 "The properties CreationClasName, SystemCreationClassName "
 "and SystemName can be set to empty strings.In this case, "
 "the CIM Object Manager must interpret the properties with "
 "the local system information.") ]
class CIM_Provider : CIM_WBEMService {
 [Override ("Name"), Description (
 "A human-readable name that uniquely "
 "identifies the provider within a system.") ]
 string Name;
 [Required, Description (
 "An implementation specific string that identifies the "
 "handle to the provider.") ]
 string Handle;
};
```

Abb. 8.4: Beispiel für das *Managed Object Format*. Die eigentliche Klassendefinition ist grau unterlegt.

8.5 CIM Schema

Das CIM Schema bietet auf Grundlage der Spezifikation ein umfassendes Datenmodell. Dieses Datenmodell besteht aus einer Menge von Klassen mit Eigenschaften und Assoziationen, die ein gut verständliches, konzeptionelles Framework bilden.

Das CIM Schema setzt sich aus dem *Core Model* und den *Common Models* zusammen, die im Folgenden genauer beschrieben werden.

8.5.1 Core Model

Das *Core Model* nimmt eine grundlegende Klassifikation der Elemente und Assoziationen vor. Alle weiteren Modelle werden von diesem abgeleitet und auf bestimmte Anwendungsfälle spezialisiert. In Abbildung 8.5 ist ein kleiner Ausschnitt des *Core Models* zu sehen, der hierarchisch gesehen die Spitze des Modells beschreibt. Die Oberklasse des Modells ist das *ManagedElement*, von der alle weiteren Klassen abgeleitet werden. [2]

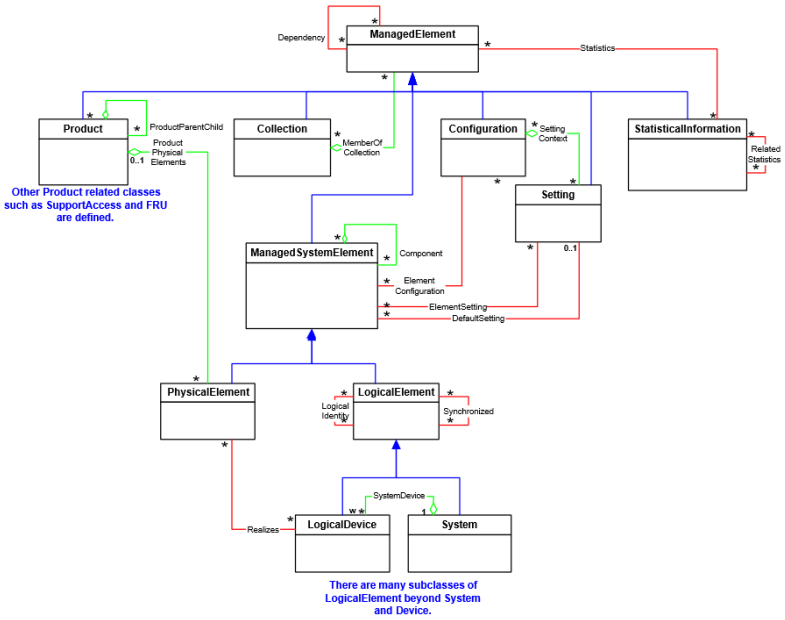


Abb. 8.5: Hierarchische Spitze des *Core Models*.

8.5.2 Common Models

Basierend auf dem *Core Model* stellen die *Common Models* eine Reihe von Informationsmodellen dar, die jeweils auf einen bestimmten Verwaltungsbereich spezialisiert sind. Dabei sind die *Common Models* unabhängig von einer bestimmten Technologie oder Implementierung. Wichtig zu verstehen ist, dass diese Modelle lediglich als Grundlagen dienen, die von Entwicklern für einen leichteren Einstieg verwendet werden können. Sie können an die eigenen Bedürfnisse angepasst werden. Gleichzeitig ist es auch möglich, keines der *Common Models* zu verwenden und stattdessen ein vollständig neues Schema zu erstellen. Das CIM Schema Version 2.7 definiert zwölf Modelle: *Applications*, *Event*, *Network*, *Support*, *Database*, *Interop*, *Physical*, *Systems*, *Devices*, *Metrics*, *Policy*, und *User*. Im Folgenden sollen beispielhaft das Application Model und das Interop Model vorgestellt werden. [2]

8.5.2.1 Application Model

Das CIM Application Management Model beschreibt Informationen, die gewöhnlicherweise für die Verwaltung von *Software Products* und *Applications* benötigt werden. Diese gliedern sich in drei Teile: Struktur, Lifecycle und Verbindungen der einzelnen Zustände im Lifecycle. Die Struktur einer Application (zu sehen in Abbildung 8.6) wird durch die Komponenten *Software Product*, *Software Feature*, *Software Element* und *Application System* definiert. Ein *Software Element* ist in diesem Kontext die kleinste Software-Einheit, die unabhängig bereitgestellt werden kann. Ein *Software Feature* ist eine Sammlung mehrerer *Software Elements*, die im Verbund eine bestimmte Funktionalität erfüllen. Analog dazu ist ein *Software Product* beziehungsweise ein *Application System* eine Sammlung verschiedener *Software Features*. Der Lifecycle umfasst das *deployment*, *installation and configuration*, *startup* und *operation including monitoring*. Aus diesen ergeben sich die Zustände *deployable*, *installable*, *executable* und *running*. Diese Zustandsinformationen werden innerhalb der *Software Elements* gespeichert. [2]

8.5.2.2 Interop Model

Im CIM Interop Model werden Verwaltungskomponenten definiert, mit denen sich die Infrastruktur des *Web Based Enterprise Management* (siehe Abschnitt 8.6) und Interaktionen mit diesem beschreiben lassen. Diese Infrastruktur ist in Abbildung 8.7 zu sehen. Die einzelnen Komponenten sind der CIM *Client*, der CIM *Server* und der CIM *Object Manager*. Der CIM Client interagiert mit dem CIM Server mittels CIM *Operation Message Requests* und CIM *Operation Message Responses*. Innerhalb des CIM Servers ist der CIM Object Manager für die Kommunikation zwischen den einzelnen Serverkomponenten verantwortlich. [2]

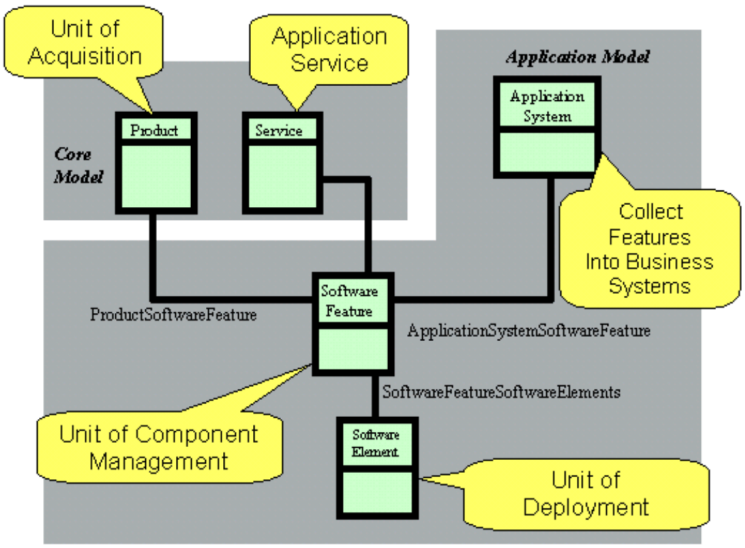


Abb. 8.6: Struktur des *Application Model* [2]

8.6 Web Based Enterprise Management

Die *Distributed Management Task Force* hat es sich zum Ziel gemacht, Standards für verteilte Desktop-, Netzwerk-, Enterprise- und Internetumgebungen zu entwickeln, um dadurch die Interoperabilität zwischen verschiedenen Management-Lösungen unterschiedlicher Anbieter zu fördern. Um dieses Ziel zu erreichen, wurden einige Standards definiert und unter dem Namen *Web Based Enterprise Management* zusammen gefasst. WBEM setzt sich aus drei Hauptkomponenten zusammen. Die erste Komponente ist das bereits vorgestellte *Common Information Model*, bei dem es sich um ein reines Informationsmodell handelt. Die zweite Komponente ist die *xmlCIM Encoding* Spezifikation, in der definiert wird, wie CIM Klassen und Instanzen durch XML Elemente repräsentiert werden können. Die dritte Komponente ist die *CIM Operations over HTTP* Spezifikation.

8.6.1 CIM XML und xmlCIM

Die xmlCIM Spezifikation beschreibt eine Grammatik, mit der XML-Dokumente erstellt werden können. Sie definiert Darstellungen für Klassen, Instanzen und Qualifier, sowie für Nachrichten innerhalb von CIM, die von CIM Kommunikationsprotokollen verwendet werden können. Für den Austausch von Informationen innerhalb von CIM sind CIM *Messages* zuständig. Es existieren verschiedene Arten von

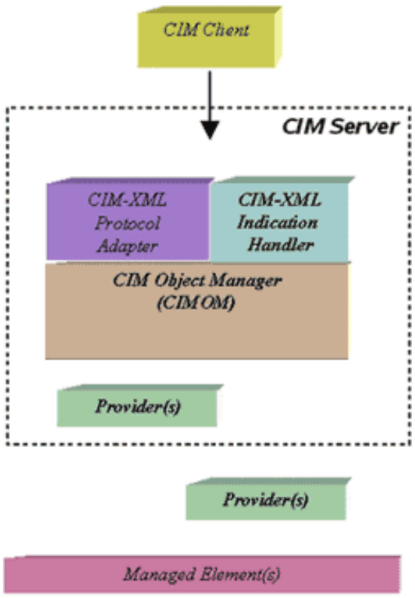


Abb. 8.7: WBEM Infrastruktur mit dem *Interop Model* [2]

Nachrichten, im einfachsten Szenario gibt es *request* und *reply* Nachrichten. Ein Grundgerüst für die Struktur einer Nachricht ist in Abbildung 8.8 zu sehen. Jeder Nachrichtentyp, der zusätzliche Informationen beinhaltet, sollte hier von abgeleitet werden. CIM Messages sind wiederum Bestandteil von CIM XML, einem standardisierten Kommunikationsprotokoll für CIM. Dieses verwendet neben der xmlCIM Spezifikation auch HTTP für den Transport. Jede Interaktion, Operation oder Datenaustausch, wird durch CIM XML in eine CIM Message verpackt, die wiederum in XML dargestellt wird. [2]

8.6.2 WBEM Operations

Die WBEM Operations spezifizieren eine Reihe von Operationen, die über HTTP ausgeführt werden können. Dabei gibt es die Möglichkeit, einzelne Operationen oder einen Stapel von Operationen sequentiell auszuführen. Ziel ist es, grundlegende, plattformübergreifende Methodenaufrufe zu ermöglichen. Zu diesen Operationen zählen zum Beispiel *getClass*, *getInstance* oder *setProperty*. Die WBEM Operations sind unabhängig von einem konkreten Protokoll definiert. Die in WBEM enthaltene CIM Operations over HTTP Spezifikation definiert eine Zuordnung dieser Operationen auf HTTP. [2]

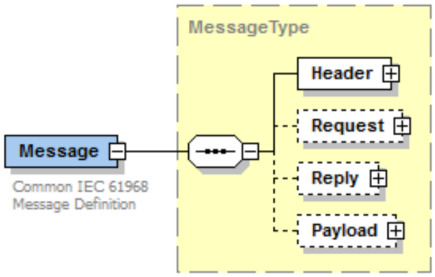


Abb. 8.8: Nachrichtenstruktur [5]

8.7 CIM in der Projektgruppe

Im Rahmen der Projektgruppe soll eine Software entwickelt werden, die mit virtuellen Kraftwerken arbeitet. Dabei werden auch einzelne Energieerzeuger simuliert. Das Smart Grid Architecture Model stellt eine geeignete Architektur dar, um solche Systeme zu strukturieren. Im SGAM gibt es unter anderem ein *Information Layer* und ein *Communication Layer*, in der die jeweiligen Standards zum Einsatz kommen. Das Common Information Model ist ein Standard, der beide Bereiche abdecken kann und wird darüber hinaus auch bei vielen Systemen, die auf SGAM aufbauen, als Standard verwendet.

Es stellt sich die Frage, welche Möglichkeiten das Common Information Model bietet, die CIM von anderen Standards unterscheidet. Wie bereits beschrieben handelt es sich beim Common Information Model um einen Informationsstandard mit einem mächtigen Datenmodell. Dieses bietet neben dem Kernmodell auch verschiedene Anwendungsmodelle, die als Grundlage für die eigene Entwicklung dienen können. Das Common Information Model wurde außerdem mit dem Ziel entwickelt, auch plattformübergreifende Systemumgebungen darstellen zu können. Unabhängig vom Informationsmodell existiert auch ein Kommunikationsprotokoll auf das CIM Basis, welches ebenfalls verwendet werden kann. Dieses Protokoll trägt die Bezeichnung CIM XML und verwendet, wie der Name bereits andeutet, XML als Darstellungsform für verschiedene Informationen.

Abschließend sollen die Vor- und Nachteile diskutiert werden, die bei der Verwendung des Common Information Models im Rahmen der Projektgruppe entstehen können. Als Gegenargument könnte aufgebracht werden, dass die Modelle von CIM teilweise sehr komplex sind und die Verwendung zusätzliche Einarbeitungszeit benötigt. Anschließend müssen ein oder mehrere Modell ausgewählt werden, die als Grundlage dienen. Diese Auswahl wird durch fehlende praktische Erfahrung mit dem Common Information Model erschwert. Ein weiteres Argument ist die Benutzung entsprechender Software, die den Umgang mit dem Common Information Model ermöglicht. Auch diese benötigt zusätzliche Einarbeitungszeit, die nicht au-

ßer Acht gelassen werden sollte. Dem gegenüber steht die Tatsache, dass in jedem Fall ein Informationsmodell der zu entwickelnden Software erstellt werden muss. Auch wenn UML den meisten Teilnehmern der Projektgruppe nicht unbekannt ist, so ist dennoch anzuzweifeln, dass die Verwendung von UML keine Einarbeitungszeit benötigen würde. Beim Common Information Model handelt es sich um einen modernen Standard, der die Erweiterbarkeit der Architektur gewährleistet, insbesondere im Hinblick auf verteilte Systemumgebungen. Nicht umsonst ist das Common Information Model ein oft in Smart Grids verwendeter Standard.

Abschließend lässt sich festhalten, dass die Nachteile weniger aus dem Modell, sondern vielmehr aus erhöhtem Aufwand resultieren. Die Vorteile hingegen entstehen aus dem Modell selbst und rechtfertigen den möglicherweise entstehenden Mehraufwand.

Literaturverzeichnis

[1] Distributed Management Task Force. *Frequently Asked Questions*. Letzer Zugriff 20.05.2015. 2015. URL: www.dmtf.org/about/faq.

[2] Distributed Management Task Force und WBEM Solutions, Inc. *CIM Tutorial*. Letzer Zugriff. 2003.

[3] Free Software Foundation Europe. *Open Standards*. Letzter Zugriff 01.06.2015. 2015. URL: www.fsfe.org/activities/os/def.en.html.

[4] Y. Tina Lee. „Information Modeling: from design to implementation“. In: (1999).

[5] M. Uslar u. a. *The IEC Common Information Model CIM*. 2012.

Kapitel 9
Programmable Logic Controller PLC und Beckhoff XAE

Torben Sauer

Zusammenfassung Programmable Logic Controllers werden seit einiger Zeit nicht nur für die Steuerung von Maschinen in der der Automatisierungstechnik eingesetzt, sondern auch im Bereich der Simulation von elektrischen Netzten. Dabei fungieren die PLCs zumeist als modellierte dezentrale Energieanlagen. Auf Basis der Modellierung von dezentralen Energieanlagen auf PLCs wurde in dieser Seminararbeit Möglichkeiten der Programmierung mit der Softwareumgebung TwinCAT von der Firma Beckhoff Automation GmbH und Co. KG und einer freien Programmierung betrachtet. Im Hinblick auf die Zielsetzung der Projektgruppe "Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks"wurde desweiteren eine Empfehlung für die Programmierung abgegeben. Als letztes wird eine Zusammenfassung und Fazit gegeben.

9.1 Einleitung

Das Simulieren von elektrischen Netzen ist einer der zentralen Bausteine in der Sicherstellung der heutigen Energieversorgung. Mit Simulationen ist es möglich Netzkomponenten optimal auszulasten, das n-1 Kriterium zu überprüfen, aber auch den Einsatz von Regelleistungen und den Handel an der Strombörse zu planen. Mit dem voranschreiten der Energiewende nimmt die ohnehin hohe Bedeutung von Netz-Simulationen weiter zu, da durch die Dezentralisierung der Energieversorgung das europäische Verbundnetz (UCTE) immer komplexer wird. Durch die vielen kleineren Anlagen und deren Einschränkungen, wie z.B. die Abhängigkeit von meteorologischen Gegebenheiten und die kleiner Erzeugungsleistung im Bezug auf die heutigen Großkraftwerke, macht es schwierig einen wirtschaftlich und technisch sinnvollen Stromhandel mit einzelnen Klein-Anlagen durchzuführen. Um diese Einschränkungen zu umgehen werden die dezentralen Anlagen zu „Virtuellen Kraft-

Carl von Ossietzky Universität Oldenburg
E-mail: torben.sauer@uni-oldenburg.de

werken“ zusammengeschlossen. Dabei handelt es sich nicht um einen physischen, sondern um einen virtuellen Zusammenschluss, der vor allem über Kommunikation der Anlagen untereinander und mit einer Leitstelle funktioniert.

Um die Virtuellen Kraftwerke (VK) zu optimieren, damit sie im Handel auch wirtschaftlich betrieben werden können, wird zum Großteil mit Simulationen gearbeitet. Auch in der Projektgruppe „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ vom Lehrstuhl Umweltinformatik der Universität Oldenburg, soll ein Portfoliooptimierer entwickelt werden, der die Einsatzplanung der Anlagen des VKs optimieren soll.

Da jedoch keine realen Anlagen im Anlagenpool eingesetzt werden, wird auf Programmable Logic Controller (PLC) zurückgegriffen. Bei diesen handelt es sich um Geräte, die zur Steuerung und Regelung von Maschinen eingesetzt werden oder zur Simulation von Energieerzeugern, wie bspw. Blockheizkraftwerken (BHKWs).

In dieser Seminararbeit sollen die Funktionsweise von PLCs im Allgemeinen betrachtet und darauf aufbauend mögliche Programmierungs-Methoden auf Basis der PLC-Software des Unternehmens Beckhoff GmbH beschrieben werden. Zudem soll am Ende dieser Arbeit eine Empfehlung für eine dieser Programmierungs-Methoden im Hinblick auf die Ausführung der Projektgruppe erfolgen.

9.2 Programmable Logic Controller (PLC)

Um eine Einschätzung über Möglichkeiten der Programmierung von Programmable Logic Controller geben zu können, soll zunächst ein Überblick über die Controller erfolgen.

9.2.1 Definition und Einsatzgebiete

Als Programmable Logic Controller definiert W. Bolton in seinem Buch „Programmable Logic Controllers“:

> „A Programmable Logic Controller (PLC) is a special form of a microprocessor-based controller that uses programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic in order to control machines and processes (vgl. [4], Seite 14).

Bei den PLCs, auch Speicherbare Programmierbare Steuerung (SPS) gennant, handelt es sich also um spezielle Mikroprozessoren, die programmierbare Speicher verwenden um Maschinen und Prozesse zu kontrollieren und zu speichern. Die PLCs werden vor allem in der Automatisierungstechnik eingesetzt und verdrängen in dieser die bisher verwendeten verdrahteten verbindungsprogrammierten Steuerungen.

Der Vorteil der PLCs ist ihr einfacher Aufbau und die Handhabung.

Die Abbildung 9.1 stellt den Aufbau der Hardware eines PLCs dar. Aus der Abbildung 9.1 kann somit entnommen werden, dass ein PLC aus einem Eingang, Ausgang, Speicher, Lade- u. Kommunikationsschnittstelle, und dem Prozessor selbst besteht. Wie in der Definition für PLCs erläutert, wird im Speicher das Programm für die Steuerung gespeichert. Die Implementation und das Monitoring des Programmes wird durch die Kommunikationsschnittstelle initialisiert.

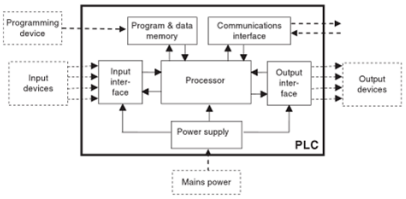


Abb. 9.1: Hardwareaufbau eines PLCs [4]

Die Ladeschnittstelle dient für die Stromversorgung. Für die Funktionsweise sind vor allem die in der Abbildung 9.1 dargestellten Eingang, Ausgang und Programm relevant. Der Eingang dient dabei zur Aufnahme von Signalen, wie beispielsweise durch Lichtschranken. Das entstandene Signal wird durch das Programm umgewandelt und dem Ausgang zugesendet. Dieser interpretiert das ankommende Signal und führt die durch das Programm ausgegebene Aktion aus. Dabei kann es sich um das Schließen eines Schaltkreises oder um das Einschalten eines Motors handeln (vgl. [7]).

Durch den, bereits erwähnten, einfachen Aufbau der Programmable Logic Controller weisen diese ein hohe Robustheit sowie eine geringe Wartungsintensität auf. Außerdem ist durch die Modularität der Zusammenschluss mehrerer PLCs möglich, was es erlaubt ganze Fertigungslinien zu steuern. Für die Programmierung werden eigens entwickelte Softwareumgebungen verwendet, die auf dem Standard IEC 61131 aufbauen. Bei dem ICE 61131 handelt es sich um das Standardprotokoll für Speicherbare Steuerungen (SPS). In diesem wird geregelt wie ein Programmcode für SPS entwickelt und aufgebaut werden soll (vgl. [12]).

9.2.2 Funktionsweise im Bereich der Virtuellen Kraftwerke am Beispiel SESA-Lab

Nachdem eine Anzahl von Einsatzmöglichkeiten und Funktionalitäten von Programmable Logic Controller betrachtet wurden, soll in diesem Unterabschnitt durch ein

Beispiel der Einsatz verdeutlicht werden. Um dem Kontext der Projektgruppe zu entsprechen, handelt es sich um ein Beispiel aus dem Bereich der Netzsimulationen.

Aus der Einleitung wurde bereits ersichtlich, dass auch im Bereich der Netzsimulationen Programmable Logic Controller verwendet werden. Bei dem gewählten Beispiel handelt es sich um die Integration eines echtzeitfähigen Labors in das Netzsimulationstool „mosaik“ des Instituts für Informatik OFFIS in Oldenburg. Für die Netzsimulationen mit mosaik können in das Tool Simulationsmodelle integriert oder aufgerufen werden. Damit die Simulationen so real wie möglich durchgeführt werden können, werden echtzeitfähige Simulatoren benötigt. Durch diese ist es möglich Daten zu jedem Zeitpunkt zu senden und zu empfangen. Des weiteren ist durch die Echtzeitfähigkeit das System mit der realen Zeit verknüpft, wodurch wiederum Anlagen verwendet werden können, die zum Zeitpunkt der Simulation ihre Tätigkeit ausführen. Mit dem Sesa-Lab ist es dem OFFIS möglich große Smart Grid Simulationen zu berechnen, sowie die Bereitstellung von Systemdienstleistungen von dezentralen Komponenten zu simulieren. Das Beispiel Sesa-Lab wurde auch im Hinblick auf die Projektgruppe ausgewählt, da ein Teil des Sesa-Labs auch in der Projektgruppe verwendet werden kann. Die Beckhoff-Komponenten werden wie in Abbildung 9.2 dargestellt als DER-Komponenten verwendet. DER bedeutet dabei „distributed energy resources“, bei denen es sich zumeist um dezentrale Energieerzeuger handelt. Diese können im SESA-Lab unterschiedliche Erzeuger darstellen, wie beispielsweise Windkraftanlagen. Die Daten dieser werden dabei in Echtzeit über die dargestellten Kommunikationsprotokolle UPC UA, IEC 61850, UML, EtherCAT oder andere empfangen und versendet. In Mosaik werden die Daten (siehe Abbildung 9.2) für Netzsimulationen verwendet und weiterverarbeitet. Durch die Echtzeitfähigkeit ist es des weiteren möglich, während der Simulation den dezentralen Energieerzeuger und Verbraucher neue Tätigkeiten zuzuweisen (vgl. [10]).

9.3 Softwareumgebung TwinCAT der Firma Beckhoff

Nachdem in Abschnitt 9.2.1 zunächst der Programmable Logic Controller und einen Beispiel für den Einsatz betrachtet wurde, soll sich nun mit den Programmierungsmöglichkeiten für PLCs befasst werden. Da in der Projektgruppe die Module des SESA-Labs verwendet werden sollen, wird im folgenden das Unternehmen vorgestellt und das von diesem angebotene Programmierungstool TwinCAT.

9.3.1 Beckhoff Automation GmbH und Co. KG

Das Unternehmen Beckhoff Automation GmbH und Co. KG wurde 1980 gegründet und hat bis heute seinen Hauptsitz in Verl inne. Die Entwicklung innovativer Produkte für PC-basierte Steuerungen ist seit jeher eines der Standbeine des Unter-

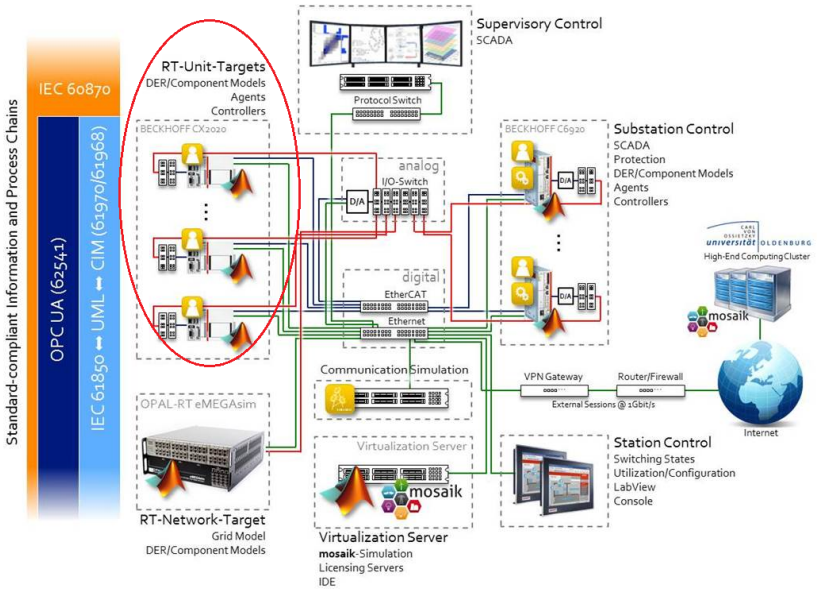


Abb. 9.2: Aufbau Sesa-Lab [9]

nehmens. Mit 2800 Mitarbeitern (Stand 04.2015) und über 70 Vertretungen Weltweit zählt Beckhoff zu einem der führenden Unternehmen in dieser Branche. Als erste Erfindungen des Unternehmens gelten das Lightbus-System und die Automatisierungssoftware TwinCAT. Da es in der Seminararbeit um die Programmierung von Steuerungsmodulen geht wird gerade diese Automatisierungssoftware im Folgenden weiter betrachtet. Als eine der neuesten Entwicklungen von Beckhoff gilt das EtherCAT, das als eine Echtzeit-Ethernet-Lösung fungiert und als ein neuer Meilenstein in der Automatisierungssoftware gilt (vgl. [2]).

Bei dem Modul EtherCAT wird das Standard Protokoll IEC 61158 verwendet. Das EtherCAT-Protokoll ermöglicht es, Synchronisationen von kleiner 1 mikro Sekunde durchzuführen, was wiederum als Echtzeit-Synchronisation beschrieben wird. Gerade die Synchronisation auf Zeitpunkte und zwischen örtlich getrennten Geräten wird so erleichtert, wodurch neue Möglichkeiten in der Automatisierungstechnik geschaffen werden. Aber auch im Bereich der Modellierung von Erneuerbaren Energieerzeugern ist die Echtzeitsimulation ein Meilenstein (vgl. [6]).

Da es aber für die Projektgruppe noch keine große Relevanz besitzt wird diese Funktion nicht weiter betrachtet.

9.3.2 TwinCAT

Die TwinCAT Serie ist wie im vorherigen Abschnitt 9.2.2 bereits angedeutet die Automatisierungssoftware der Firma Beckhoff. Das neueste Produkt der Reihe ist das TwinCAT 3. TwinCAT steht für „The Windows Control and Automation Technologie“ und stellt eine PC-basierte Automatisierungssoftware dar, die eine große Anzahl an Quellcodes in Echtzeitsteuerung umwandeln kann. TwinCAT verwendet Microsofts Visual-Studio® als Plattform für die Programmierung, des weiteren baut diese auf dem ICE 61131 auf und ist frei zugänglich (vgl. [2]).

TwinCAT bietet eine Vielzahl von Möglichkeiten zur Erstellung von Programm-

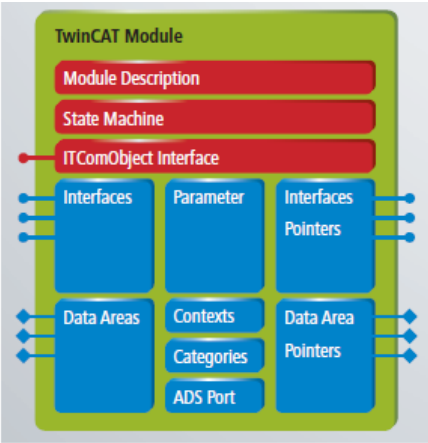


Abb. 9.3: Aufbau TwinCAT Modul [2]

codes, da jedoch nicht alle für die Projektgruppe relevant sind, werden im weiteren Verlauf nur relevante Tools betrachtet. Für die Programmierung mit der Automatisierungssoftware werden vier unterschiedliche Module verwendet.

- eXtended Automation Architecture (XAA)
- eXtended Motion Control und Play Safe
- eXtended Automation Engineering (XAE)
- eXtended Automation Runtime (XAR)

Bei dem Modul XAA handelt es sich um die Softwarearchitektur des TwinCAT-Systems, die es ermöglicht die Programmierung mit dem ICE 61131, aber auch die Programmierung über andere Sprachen, vor allem mit C++ und Matlab/Simulink durchzuführen. Genauso ist die Integration bereits vorhandener Automatisierungstools in TwinCAT über die Architektur möglich.
Bei eXtended Motion Control handelt es sich um ein Modul für durchgängige und

skalierbare Lösungen in der Bewegungssteuerung. Durch das Modul ist eine Punkt-zu-Punkt Bewegung, sowie die Ansteuerung von Robotern möglich. Das Modul Play Safe ist für die Sicherung der Applikationen zuständig, aber auch für die Überprüfung der Konsistenz in den Quellcodes.
Das Module XAE beinhaltet die Engeniering-Umgebung von TwinCAT indem mit unterschiedlichen Programmiersprachen, Programme entwickelt werden können.
Bei dem letzten wichtigen zu betrachtende Modul handelt es sich um das Modul XAR. Dieses stellt eine Umgebung dar, in der fast alle mit dem XAE erstellten Programme ablaufen können. Zu dem bietet die Umgebung den Realtimekontext.

Da gerade die Module XAE und XAR eine besondere Relevanz für die Projektgruppe besitzen, werden diese in den nächsten Abschnitten 9.3.3 ausführlich betrachtet. Neben den Modulen bietet auch der Aufbau des eigentlichen programmierten Systems in TwinCAT durch das XAE Vor-und Nachteile für die Modellierung eines Energieerzeugers. Damit diese auch in die Betrachtung der Möglichkeiten mit einbezogen werden können, werden auch hier die wichtigsten erläutert. Die Abbildung 9.3 (vgl. [3]) zeigt den Standardaufbau einer programmierten Software mit XAE, dabei müssen vor allem zwei Punkte genauer betrachtet werden Statemachine und die Kommunikationsschnittstelle. Beide können für die Programmierung als Restriktionen gelten, weshalb sie für die Betrachtung der Programmierungsmöglichkeiten im Bezug auf die Projektgruppe besonders interessant sind.

Jedes mit XAE entwickelte Programm implementiert eine Statemachine, welche die Initalisierung, Parametrierung und Verlinkung der Module steuert. Damit ist es ein zentraler Baustein in der Programmierung mit XAE. Jede Statemachine besteht dabei aus vier Zuständen, die aus Abbildung 9.4 entnommen werden können. In den vier Zuständen ist die Daten-Kommunikation unterschiedlich möglich. Im Init-Zustand ist weder die Mailbox- noch die Datenprozesskommunikation möglich. Nach der Überprüfung erfolgt der Wechsel in Zustand zwei, der mit Pre-Op benannt wird, in diesem ist zunächst die Mailbox-Kommunikation freigeschaltet und die Prozessdatenübertragung wird eingestellt. Bevor der Zustand Safe-Op erreicht wird, werden alle Inputdaten kopiert und verteilt. In diesem Zustand ist nun auch die Prozessdatenkommunikation möglich. Jedoch werden die nun vorhandenen Outputdaten noch nicht ausgegeben. Die Outputdaten werden vor dem Erreichen des letzten Zustands Op wiederum kopiert und verteilt. Auch in diesem Zustand sind beide Kommunikationswege möglich (vgl. [6]).
Für die Betrachtung der Programmierungsmöglichkeiten für die Projektgruppe ist das Vorgehen der Statemachine im TwinCAT-Modul in sofern von Bedeutung, dass die Programmierung genau auf dieses Vorgehen abgestimmt sein muss. Dies kann zum einen durch das klare Vorgehen ein Vorteil für die Programmierung sein, jedoch auch als starke Einschränkung und somit als Nachteil gesehen werden. Da die Programmierung mit einer Statemachine für die Projektgruppe zunächst eine neue Art der Programmierung darstellt, wird im folgenden die Statemachine als Restriktion

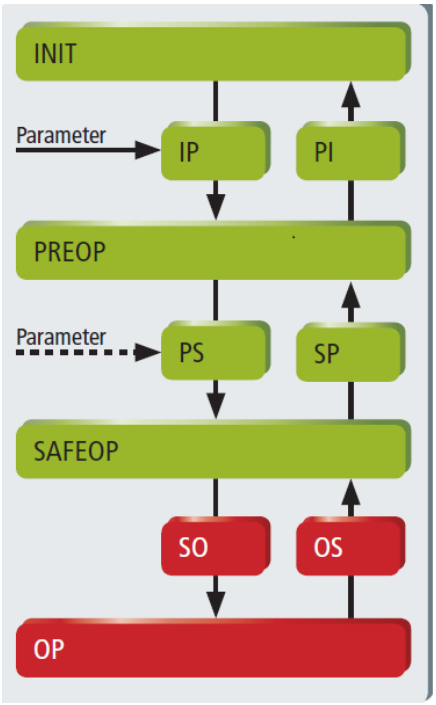


Abb. 9.4: Statemachine im Kontext von TwinCAT [2]

und somit als eines der Nachteile der TwinCAT-Programmierung angesehen.

Neben der Statemachine kann auch die Auswahl des Kommunikationsprotokolls für die Kommunikation zwischen Modulen und anderen Systemen kann als eine Restriktion bzw. Begrenzung angesehen werden, da TwinCAT beispielsweise nicht alle vorhandenen Protokolle unterstützt. Jedoch ist diese Restriktion nicht überzubewerten, da bei der Verwendung auf die Vorhanden und bereits implementierten Kommunikationsprotokolle zurückgegriffen werden können.

9.3.3 eXtended Automation Engineering (XAE)

Wie bereits angedeutet, handelt es sich bei dem XAE um das Engineering Modul in TwinCAT. Es ermöglicht Automatisierungsobjekte parallel mit Hilfe von unterschiedlichen Programmiersprachen zu erzeugen. Dabei ist es möglich, dass die Objekte, egal in welcher Sprache sie programmiert wurden, sich gegenseitig aufrufen und miteinander kommunizieren können. Die Möglichkeiten für die Programmierung mit dem XAE-Modul können der Abbildung 9.5 entnommen werden.

Diese zeigt eine Vielzahl von Möglichkeiten, die jedoch nicht alle in dieser Seminararbeit betrachtet werden können. Da in der Projektgruppe keine Speicherbare Steuerung für die Automatisierungstechnik programmiert werden soll, fällt die Programmierung über das IEC 61131 aus der Betrachtung der Möglichkeiten heraus. Desweiteren ist es auf den ersten Blick als sinnvoll zu erachten eine Programmierung zu verwenden, die mit dem XAR Modul kompatibel ist, damit man von den dessen Vorteilen profitieren kann. Um welche Vorteile es sich handelt wird im folgenden Abschnitt 9.3.4 beschrieben. Durch diese Einschränkungen wurden 4 Varianten identifiziert, die für die Projektgruppe interessant sein könnten (vgl. [3]).

- Implementierung in XAE, im Realtime Kontext auf Basis Matlab/ Simulink
- Implementierung in XAE, im Realtime Kontext auf Basis C/ C++ oder IEC 61331
- Implementierung innerhalb XAE, im Non-Realtime Kontext, mit C sharp oder .NET
- Freies Simulationsmodell ohne Verwendung des XAE

Für die Zielerfüllung einer Einschätzung der möglichen Programmierungsvarianten

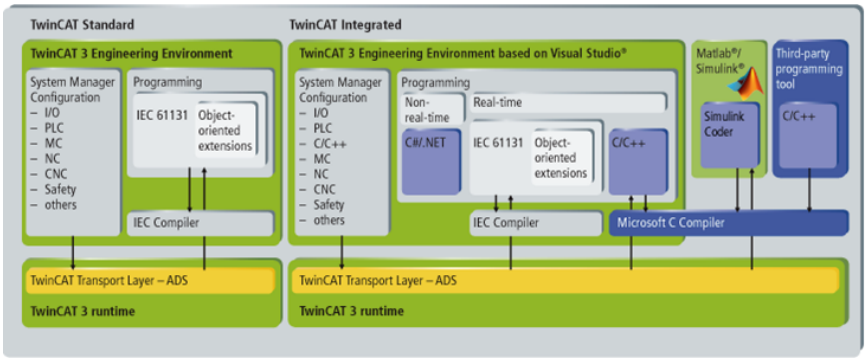


Abb. 9.5: Programmierungsmöglichkeiten in TwinCAT [3]

sollen diese in eigenen Abschnitten 9.4 betrachtet werden. Dafür findet eine Betrachtung von Vor- und Nachteile bzw. Einschränkungen in der Implementierung statt. Zunächst jedoch soll mit dem XAR ein weiteres wichtiges Modul von TwinCAT beschrieben werden.

9.3.4 eXtended Automation Runtime (XAR)

Auch bereits grob beschrieben, wurde das Modul eXtended Automation Runtime (XAR), welches in TwinCAT die Umgebung darstellt in der Programme ablaufen können. Dabei ist es jedoch nicht unerheblich, welche der ausgewählten Varianten

verwendet wird, da nicht alle Varianten, die TwinCAT verwenden, auf dieses Modul zurückgreifen können. Dies wird aber in den folgenden Abschnitten für jedes Beispiel erläutert. XAR dient als offene Kommunikationsschnittstelle zwischen den einzelnen Bausteinen. Dabei dient ADS als Hauseigenes Kommunikationsprotokoll. Durch das XAR ist es den Modulen, die sowohl komplex, als auch einfach aufgebaut sein können, möglich auf Echtzeitbasis zu interagieren. Das XAR ist somit ein wichtiges Tool bei der Programmierung mit TwinCAT, da durch die Bausteinlösung jeder Abschnitt separat entwickelt werden kann, wodurch die Komplexität eingeschränkt wird und für die einzelnen Abschnitte ein hoher Wiederverwendungswert entsteht (vgl. [2]).

9.4 Möglichkeiten der Modellierung eines Simulationsmodells

Nachdem im vorherigen Abschnitt 9.3.2 die Automatisierungssoftware TwinCAT, zusammen mit den wichtigsten Bausteinen vorgestellt wurde, soll in diesem Abschnitt die Möglichkeiten der Programmierung mit und ohne TwinCAT betrachtet werden.

9.4.1 Implementierung in XAE auf Basis Matlab/ Simulink

Die erste Möglichkeit, die betrachtet wird, ist die Programmierung in XAE mit Matlab/ Simulink. Diese Programmiersprache wurde in den 1970er Jahren als kommerzielle Software von der Firma MathWorks entwickelt. Bei Simulink handelt es sich um ein Erweiterungstool von Matlab, das speziell für die Programmierung von physikalischen, technischen und finanzmathematischen Systemen verwendet wird. Dabei wird bei der Entwicklung von Systemen auf das Baukastenprinzip zurückgegriffen, viele Komponenten können somit durch vorgefertigte Blöcke dargestellt werden. Wie in Abbildung 9.6 zu entnehmen, kann man mit Simulink auch „graphisch“ programmieren: durch das Setzen und Verbinden der einzelnen Blöcke wird im Hintergrund der Code generiert. Das Beispiel zeigt die Modellierung eines Stromkreises sowie eines Ankerkreises.
Die entwickelten Simulink-Programme können nach ihrer Fertigstellung in das XAE von TwinCAT hochgeladen werden, wo die Umwandlung in ein C-Programm stattfindet. Jedoch ist dies nicht ohne Restriktionen möglich, auf die auch schon während der Entwicklung geachtet werden muss. Dabei ist zum einen die Bausteinentwicklung selbst zu nennen, diese kann zum einen dabei helfen komplexe Systeme einfach darzustellen und dient dabei noch als sehr schnelle Programmierung, „Rapid Prototyping“. Jedoch kann diese auch hinderlich sein, da auf die Restriktionen Statemachine und auch die verwendbaren Kommunikationsprotokolle, die für XAE von Bedeutung sind, geachtet werden muss. Des weiteren ist durch die Bausteine die Programmierung soweit vorgegeben, dass auf die vorhandenen Bibliotheken, die in TwinCAT

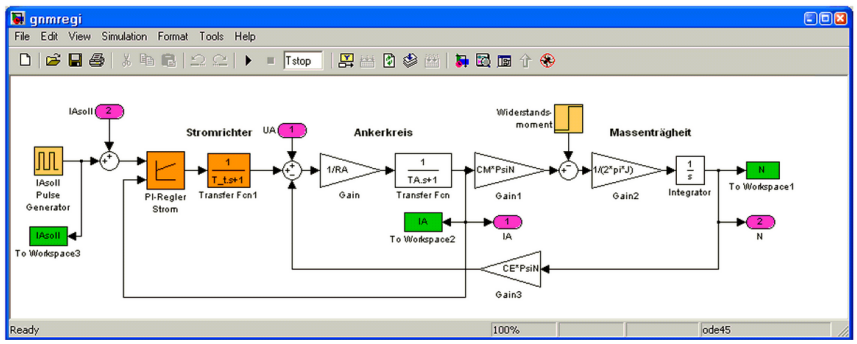


Abb. 9.6: Beispiel Matlab/Simulink [1]

implementiert wurden, zurückgegriffen werden muss. Für die Projektgruppe bedeutet die Programmierung mit Matlab/ Simulink zum einen die Programmierung auf Basis von vorgefertigten Bausteinen/ Toolboxen unter den Restriktionen von XAE. Zum anderen kann bei dieser Möglichkeit die Runtime von TwinCAT genutzt werden, welches das XAR beinhaltet. Somit kann auf das Debugging, die Echtzeitfähigkeit der Module, das ADS-Kommunikationsprotokoll und auf die Beckhoff-Dokumentation zurückgegriffen werden. Als weiteren Vorteil bietet Simulink eine Vielzahl von Toolboxen und Beispielen (vgl. [1]).

9.4.2 Implementierung in XAE, im Realtime Kontext auf Basis C/ C++ oder IEC 61131

Auch bei der Programmierung mit C++ kann auf den Realtime Kontext von TwinCAT zurückgegriffen werden, mit allen Vorteilen, die bereits in dem Unterabschnitt 9.4.1 vorgestellt wurden.

Die Programmiersprache C++ ist eine ISO genormte Programmiersprache, die als objektorientierte Erweiterung der Sprache C entwickelt wurde. Dabei handelt es sich um eine mächtige Sprache, die in der maschinennahen Programmierung häufig eingesetzt wird. Im Gegensatz zu Matlab/ Simulink wird nicht auf graphische Programmierung und das Baukastenprinzip gesetzt. C++ basiert wie C auf nur wenigen Schlüsselwörtern und bezieht ihre meiste Funktionalität aus Bibliotheken und Frameworks. Daraus ist ersichtlich, dass auch die Programmierung mit C++ auf vorgefertigten Code und Beispielen basiert, jedoch kann dies für die Integration in die TwinCAT-Software ein Hindernis darstellen. Für die Kompilierung des Programms in XAE dürfen keine Dritt-Bibliotheken für C++ verwendet werden, damit sind alle Bibliotheken gemeint, die nicht zu den Standardbibliotheken von C++ gehören. Des weiteren ist es nicht möglich, ein Programm auf Basis von C++ zu verwenden, wel-

ches den Befehl new() enthält oder dynamische Bibliotheken (DLL) aufruft.

Die auf dem IEC 61331 basierende Programmierung kann in jeder Softwareumgebung erfolgen, die diesen Standard verwendet. Solange der Standard eingehalten wird, ist die Integration in XAE möglich. Da diese Programmierung jedoch auf die Steuerung von Maschinen in der Fertigung spezialisiert ist und auch eine Vielzahl der Software kostenpflichtig ist, fällt diese Möglichkeit aus der Betrachtung auf das Ziel der Projektgruppe heraus (vgl. [7]).

Die Programmierung mit C++ besitzt damit also auch die Vorteile, die TwinCAT mit dem XAR anbietet und darüber hinaus die eigenen Vorteile der mächtigen Programmiersprachen bzw. der maschinennahen Programmierung, wird aber gleichzeitig durch die Verwendung der Standardbibliotheken eingeschränkt (vgl. [11]).

9.4.3 Implementierung innerhalb XAE, im Non-Realtime Kontext, mit C sharp oder .NET

Bei der Programmierung mit C sharp oder .Net handelt es sich um die dritte Möglichkeit und die letzte zu betrachtende in XAE. Die Besonderheit dieser Möglichkeit ist, dass sie nicht im Realtimekontext stattfindet. Das bedeutet, wie in Abbildung 9.5 dargestellt, kann diese Möglichkeit nicht auf das XAR Modul zurückgreifen und besitzt somit auch nicht die Vorteile dessen.

Die Firma Microsoft veröffentlichte mit .NET seine eigene Programmierungsumgebung, die nur für Windows verfügbar ist. Die verwendete Plattform wird durch das Visual Studio dargestellt (vgl. [8]). C sharp ist eine für diese Umgebung entwickelte Programmiersprache, die auf Sprachen wie C++ und Java basiert, weshalb sie auch zu den objektorientierten Programmiersprachen zählt. Wie C++ besitzt C sharp eigene Klassenbibliotheken und verfügt über die .Net-Komponenten eigene Schnittstellen (vgl. [5]).

Da auch die .Net-Umgebung auf das Visual Studio zurückgreift, ist dieses einfach mit dem TwinCAT-Modul zu verknüpfen, jedoch können Schwierigkeiten in der Ausführung bestehen, da die .Net-Umgebung eine eigene Laufzeitumgebung und Kommunikationsprotokolle verwendet. Auch in diesem Fall müssen Restriktionen wie die Statemachine beachtet werden. Auch die Dokumentation der Integration in XAE ist durch Beckhoff nicht in der Ausführlichkeit behandelt, wie es bei Simulink und C++ der Fall ist.

9.4.4 Freies Simulationsmodell ohne Verwendung des XAE

Nachdem die Möglichkeiten der Programmierung für das XAE-Modul betrachtet wurden, soll in einem letzten Schritt die Möglichkeit der freien Programmierung betrachtet werden. Dies bedeutet, dass komplett unabhängig von der TwinCAT-Software programmiert werden kann und die vorhandenen PLCs lediglich als Rechner (Koppler) angesehen werden können. Somit wäre es für die Projektgruppe freigestellt, welche Programmiersprache und -umgebung verwendet werden soll. Zudem ist auch die Wahl der Kommunikationsprotokolle der Projektgruppe überlassen. Nachteile liegen in dieser Variante vor allem darin, dass die Vorteile des XAR- und XAE-Moduls, wie beispielsweise das Debugging, die Echtzeitfähigkeit und die Kommunikationsschnittstellen, nicht verwendet werden können. Dafür muss bei der freien Programmierung nicht auf die Restriktion Statemachine geachtet werden. Bei dieser Variante spielt vor allem das vorhandene Know-How eine wichtige Rolle.

9.5 Empfehlung einer Möglichkeit im Hinblick auf die Projektgruppe

Zusammenfassend zeigt die Programmierung mit dem Beckhoff XAE Vorteile im Bereich des Zusammenspiels der einzelnen Module, gerade mit dem Modul XAR. Dadurch kann auf vorgefertigte Bausteine für die Entwicklung, Debugging, Kommunikationsschnittstellen usw. zurückgegriffen werden, welche zudem miteinander kompatibel sind. Wird sich also bei den Möglichkeiten der XAE-Programmierung an das Vorgehensmodell gehalten, sollte es zu keinen Problemen bei der Implementierung auf den PLCs, sowie bei der Kommunikation zwischen Systemen kommen. Des weiteren bietet die Programmierung mit XAE den Vorteil der bereits erläuterten Echtzeitsimulation.
Neben den Vorteilen gibt es jedoch auch einige Einschränkungen, die durch die Projektgruppe beachtet werden müssen, wenn sich für eine Möglichkeit mit XAE entschieden wird. Dabei stehen vor allem die Einschränkungen von Statemachine, Auswahl der Kommunikationsprotokolle und die Verwendung von Standardbibliotheken bzw. der Programmierung auf Basis des Baukastenprinzips im Vordergrund. Auch für die freie Programmierung gibt es sowohl Nachteile als auch Vorteile, weshalb eine eindeutige Empfehlung sehr schwierig ist.

Da gerade der Einsatz der TwinCAT-Umgebung viele Vorteile bietet, wenn man den Realtime-Kontext verwendet, fällt die Empfehlung auf die Verwendung von Matlab/-Simulink oder C++. Welche dieser beiden Möglichkeiten am Ende ausgewählt wird oder ob doch eine der anderen Möglichkeiten zum Einsatz kommt, hängt vor allem von dem Know-How der Projektgruppe und den Anforderungen der Auftraggeber ab. Die größten Vorteile, die die Programmierung mit den empfohlenen Varianten

bietet, ist die ausführliche Dokumentation für TwinCAT und die vielen vorhandenen Beispiele der Simulation von dezentralen Energieerzeugern.

9.6 Fazit

Bei den PLCs handelt es sich um Programmierbare Speicher, die sowohl in der Automatisierungstechnik, als auch in der Simulierung von dezentralen Energieerzeugern im Kontext der Netzsimulationen und Virtuellen Kraftwerken eingesetzt werden. Die Möglichkeiten der Programmierung der PLCs sind vielfältig, gerade die Firma Beckhoff bietet mit ihrem TwinCAT eine Softwareumgebung mit einer Vielzahl von Möglichkeiten der Programmierung von PLC-Programmen an. Durch die vorhandenen Beckhoff-Module, die auch Einsatz im SESA-Lab des OFFIS in Oldenburg finden, ist die Verwendung dieser auch für die Projektgruppe „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ der Universität Oldenburg sinnvoll.Bei der Betrachtung von 3 Möglichkeiten der Programmierung mit dem Beckhoff XAE und einer freien Programmierung ohne Beckhoff wurde in dieser Seminararbeit eine Empfehlung für das XAE ausgegeben. Gerade die Möglichkeiten der Programmierung mit Matlab/Simulink oder C++ bieten in diesem Zusammenhang viele Vorteile. Da diese Seminararbeit jedoch das Ziel der Beschreibung der Möglichkeiten der Programmierung hat und das Vorwissen der Projektgruppenteilnehmer in die Betrachtung nicht eingeflossen ist, kann zu diesem Zeitpunkt noch keine eindeutige Empfehlung abgegeben werden.

Literaturverzeichnis

[1] A. Angermann u. a. *Matlab - Simulink - Stateflow*. 8. Aufl. De Gruyter Oldenbourg Verlag, 2014.

[2] Beckhoff. *TwinCAT SPS und Motion Control auf dem PC*. Die Beckhoff Automation GmbH & Co. KG. 2015.

[3] BeckhoffXAE. *TwinCAT 3 | eXtended Automation (XA)*. 2015. URL: http://www.beckhoff.de/default.asp?twincat/twincat-3-xa-language-support-c.htm?id=1893323818933256.

[4] W. Bolton. *Programmable Logic Controllers*. 6. Aufl. Newnes Verlag, 2015.

[5] ECMA. *C sharp Language Specification*. emac International 334. 2006.

[6] EtherCAT Technology Group. *EtherCAT - the Ethernet Fieldbus*. 2003. URL: http://www.ethercat.org/en/technology.html.

[7] D. E. Kendry. *Programmable Automation Technologies*. 1. Aufl. Industrial Press, 2010.

[8] Microsoft. *Visual Studio*. 2015. URL: https://msdn.microsoft.com/de-de/vstudio/.

[9] OFFIS. *SESA-Lab*. 2014. URL: http://www.opal-rt.com/success-story/greener-smarter-better-connected-grids-innovation-sesa-lab.

[10] Sebastian Rohjans. „Smart Grid CoSimulation with mosaik and the SESA-Lab“. In: OFFIS Institut für Informatik, 2014.

[11] H. Tschabitscher. *Einfuehrung in C++-Programmierung*. http://ladedu.com/cpp/. 1997.

[12] TUDresden. „DIN EN 61131 Ein abstraktes, vereinheitlichendes Modell einer speicherprogrammierbaren Steuerung (SPS)“. 2012.

Kapitel 10
PG „HW-basierte Simulation “ + OPC UA

Marco Schnieders

10.1 Einleitung / Motivation

Die Projektgruppe Ecob stellte sich aufgrund der steigenden Anzahl der dezentralen Energieerzeugungsanlagen in Form von Photovoltaikanlagen, Blockheizkraftwerken und ähnlichen Anlagen in den letzten Jahren der Aufgabe, ein System zu entwickeln, das den Einsatz steuerbarer Energieerzeuger, -verbraucher und -speicher hinsichtlich der Minimierung von Energiebezugskosten optimiert. [6]

In dieser Seminararbeit wird rudimentär dargelegt, mit welchen Themen sich die PG Ecob aus dem Bereich der hardwarebasierten Simulation von Energieanlagen auseinandergesetzt hat. Zudem werden Eindrücke, Empfehlungen, das Fazit und der Ausblick der Projektgruppe zusammengefasst, um der chronologisch und thematisch anschließenden Projektgruppe „VK“ ein Basiswissen zur Verfügung zu stellen, mit dessen Hilfe frühe und auch im weiteren Verlauf des Projekts wichtige Entscheidungen unterstützt bzw. positiv beeinflusst werden sollen. Es stellt sich weiterhin die Frage, wo die Gemeinsamkeiten und die Unterschiede in der Thematik zwischen den beiden Projektgruppen liegen und welche Motivation der neuen PG VK zugrunde liegt. Abschließend wird erörtert, inwieweit sich Teile der Ergebnisse oder des Produkts der PG Ecob für die PG VK wiederverwenden lassen.

10.2 Die Projektgruppe Ecob

Die Motivation der Projektgruppe Ecob bzw. „Hardware-basierte Simulation energieautonomer Gebäude“ ergibt sich aus ihrer Vision. Die Vision zeigt auf, dass ein Energiemanagementsystem für ein Simulationssystem entwickelt werden sollte, das aus mehreren Komponenten besteht. Komponenten können Energieerzeuger, Ener-

Carl von Ossietzky Universität Oldenburg
E-mail: marco.schnieders@uni-oldenburg.de

giespeicher und Energieverbraucher sein. Die Simulation soll sich auf ein Gebäude beschränken, das durch Kenndaten über die elektrische und thermische Energie, die Gebäudegröße, die technischen Eigenschaften der Komponenten sowie die Wetter- und Klimabedingungen definiert ist. Für die Simulation sollen sich beliebige Szenarien erstellen und durchführen lassen, die am Ende eines Durchlaufs einen Fahrplan für die im Szenario definierte Anlage bereitstellt. Das Ziel ist, mit Hilfe des Systems einen Fahrplan zu berechnen, der den Eigenverbrauch des energieautonomen Gebäudes optimiert, um die Energiebezugskosten im Rahmen der Simulationsdauer zu minimieren. Zudem soll das Ergebnis zu evaluieren sein, indem geprüft wird, ob die Anlage im Vergleich zu einem ungesteuerten Fahrplan mehr oder weniger Energie aus dem öffentlichen Netz bezogen hat. Außerdem werden die Ergebnisse verschiedener Szenarien dahingehend verglichen, welche Parameteränderungen positive Ergebnisse erzielt haben. [6]

Um gedanklich in das Themenfeld einzusteigen, befasst sich der folgende Abschnitt 10.2.1 mit dem Energienetz, das alle die elektrische Energie betreffende Bestandteile regional und überregional vernetzt und an das auch die dezentralen Energieerzeugungsanlagen angeschlossen sind.

10.2.1 Das Energienetz

Das Netzwerk in der elektrischen Energietechnik wird in der Regel als Stromnetz bezeichnet und besteht aus elektrischen Stromleitungen wie Freileitungen und Erdkabeln, Netzkomponenten wie Schalt- und Umspannwerke, die unter anderem den Strom zwischen unterschiedlichen Spannungsebenen transformieren, sowie daran angeschlossenen Energieerzeuger (Kraftwerke) und Verbraucher. Unser Energienetz ist ein Wechselspannungsnetz, das bei 50Hz arbeitet. In diesem Netz existieren verschiedene Spannungsebenen, je nach Anwendungszeck des Stroms. Das Höchstspannungsnetz und das Hochspannungsnetz übertragen Strom über große Distanzen bei 400kV und 230kV bzw. 110kV und werden zusammen als Transportnetz bezeichnet, während das Mittelspannungsnetz und das Niederspannungsnetz als Verteilnetz bezeichnet werden. Das Mittelspannungsnetz überträgt den Strom bei 1kV bis 30kV und das Niederspannungsnetz überträgt mit 230V bzw. 400V. An die beiden zuletzt genannten Netze sind in der Regel die Endverbraucher angebunden. Die großen Energieerzeuger sind in der Regel direkt an das Verteilnetz angebunden. Zwischen den Netzen wird die Stromspannung mittels Transformatoren umgewandelt. Dies verringert den Leistungsverlust, vor allem bei der Übertragung über große Strecken. Dort wird die Spannung erhöht, um den für die Übertragung benötigten Leitungsquerschnitt und die Ohm‘schen Verluste so gering wie möglich zu halten. Generell geht ein Teil der elektrischen Energie bei der Übertragung unter anderem in Wärmeenergie über. [9]

Die folgende Grafik vermittelt einen Überblick über die verschiedenen Spannungsebenen:

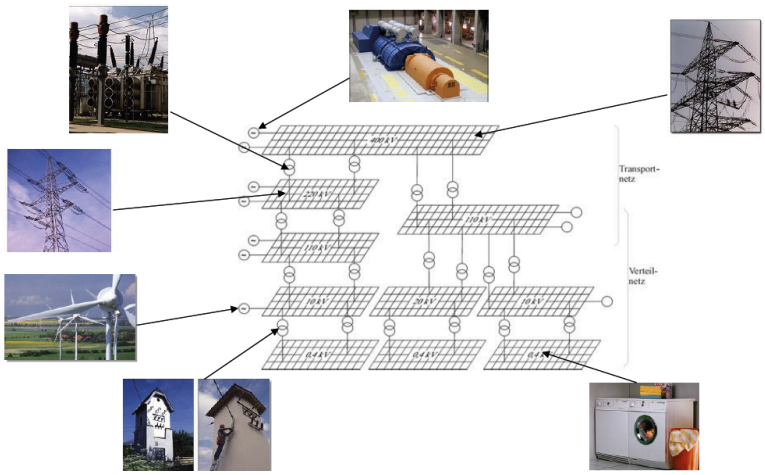


Abb. 10.1: Überblick Stromnetz [2]

Durch den großen Zuwachs an dezentralen Energieerzeugungsanlagen wie Photovoltaikanlagen, Blockheizkraftwerke oder Windkraftwerke in den letzten Jahren wird das Nieder- und Mittelspannungsnetz (Verteilnetz) stark belastet, da diese kleinen „Kraftwerke“ auf diesen Ebenen in das Stromnetz einspeisen. Es gibt wesentlich mehr Knotenpunkte im Netz und eine stark fluktuierende Einspeisung beeinflusst das Gleichgewicht zwischen Verbrauch und Erzeugung und verursacht damit auch Frequenzschwankungen im Netz, die ständig ausgeglichen werden müssen, um das Netz stabil zu halten. [6]

Dementsprechend ist es sinnvoll, nach Lösungen zu suchen, die der weiteren Belastung des Stromnetzes entgegenwirken oder das Stromnetz an anderer Stelle wiederum entlasten. Als elementarer Gesichtspunkt wird von Institutionen wie dem Bundesministerium für Wirtschaft und Energie (BMWi) und dem Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit (BMUB) dabei der Ansatz gesehen, den Strom dort zu verwenden, wo er erzeugt wird. Das heißt unter anderem, dass energieautonome Gebäude zunächst den eigenen Energiebedarf decken sollen und nur überschüssige Energie in das Stromnetz eingespeist werden soll. [1]

Durch diesen sogenannten Eigenverbrauch können in der Theorie die Energiebezugskosten, also die Kosten, die für den altbekannten Bezug von Strom aus dem Stromnetz anfallen, gesenkt werden. Da jedoch bei einem Gebäudeeigenen Energieerzeugungs-, Speicherungs- und Verbrauchssystem auch Bau-, Wartungs- und Betriebskosten anfallen, ist es möglich, dass es unter bestimmten Umständen günstiger ist, den Strom trotzdem herkömmlich aus dem Stromnetz zu beziehen und die Gebäudeeigene Anlage um ein gewisses Level oder sogar ganz herunterzufahren.

Die Energieproduktion einer Gebäudeeigenen Energieerzeugungsanlage hängt von ihren Produktionsfaktoren ab. Je nachdem, um welche Anlagenkomponenten es

sich handelt, können dies Faktoren wie Windstärke, Sonneneinstrahlung oder auch Roh- und Kraftstoffpreise bzw. dessen Verbrauch sein.

Der nächste Abschnitt 10.2.2 erläutert das weitere Vorgehen der Projektgruppe nach Abschluss der Einarbeitungsphase.

10.2.2 Problemanalyse

Der Problemanalyseprozess diente dazu, die Vorstellungen aller am Projekt beteiligten Parteien auf einen gemeinsamen Nenner zu bringen, indem das Problem zunächst genau erläutert, Akteure und Systeme identifiziert und Anwendungsfälle erarbeitet wurden. [6]

Die Problembeschreibung umfasste, dass ein System zur Steuerung von Energieerzeugern (Blockheizkraftwerke und Photovoltaik), -speichern und -verbrauchern eines simulierten Gebäudes in einer simulierten Umgebung zu realisieren sei. Teilaspekte davon waren zudem das Entwickeln eines Steuersystems, das Simulieren eines Gebäudes für die Anwendung und das Testen des Steuersystems, sowie das Definieren der Zielvorgaben der Steuerung und des Umfangs bzw. Detailgrads der Simulation. Erfasst wurden die Informationen zur Problemstellung durch die Durchführung von Workshops. [6]

Das Ergebnis der Problemanalyse war gleichzeitig die eindeutige Formulierung des Ziels für die Projektgruppe. Zusammengefasst bedeutete dies die Entwicklung eines Energiemanagementsystems (EMS) für ein aus mehreren Komponenten bestehendes Simulationsframework (SF), für welches das EMS eine Eigenverbrauchsoptimierung durchführt. Simulationsframework-Komponenten sind Energieerzeuger, -speicher und -verbraucher als Teil eines simulierten Gebäudes in einer simulierten Umgebung. [6]

Diese beiden Systembauteile, das EMS und das SF, bilden zusammen mit einer Steuerungsebene (SE) das Gesamtsystem. Die Steuerungsebene dient den Akteuren im System als Benutzerschnittstelle für das EMS und/oder SF. Dies hat den Grund, dass das EMS das eigentliche Produkt des Projekts sein sollte und möglichst unabhängig funktionieren sollte. Das EMS muss jedoch natürlich mit Daten und Einstellungen versorgt werden, welche zum Teil auch händisch von Benutzern eingegeben werden sollen. Auch der Start und Stopp der Simulation wird über die Steuerungsebene kontrolliert. Zudem kann sie auch die Daten und Ergebnisse für den Benutzer visualisieren. [6]

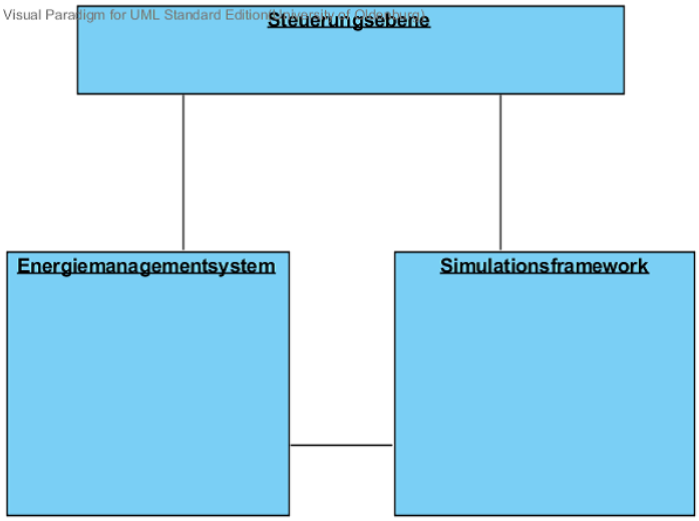


Abb. 10.2: Abstrakte Abbildung des Gesamtsystems Ecob [6]

Weitere Details der Ergebnisse der Problemanalyse werden im folgenden Abschnitt 10.2.3 erläutert.

10.2.3 Simulation eines autonomen Energiesystems

Das Projekt „Hardwarebasierte Simulation energieautonomer Gebäude“, das von den Mitgliedern der Projektgruppe später in „Ecob“ umgetauft wurde, lenkte das gesamte Projekt von vornherein in Richtung Simulation. In einer Simulation wird ein Modell der Realität nachgebildet und daran experimentiert. [8]

Bevor die Projektgruppe mit der Entwurfsphase startete, konnte die Aufgabenstellung wie folgt umrissen werden:

> Simulation eines autonomen Energiesystems (Erzeugung, Speicher, Verbrauch) mit frei konfigurierbaren Produktionsfaktoren (Wind, Sonne, Roh-/Kraftstoff(-preis)) und vorkonfigurierten Energieerzeugern, -speichern und -verbrauchern auf Basis realer Eigenschaften bzw. Daten für verschiedene Gebäudetypen mit definierten Kenndaten (Grundlast, Dimensionierung/Anzahl der Energieanlagenteile, ...)

Die Projektgruppe musste zudem ermitteln, mit welchen Daten das System später arbeiten würde und ein Datenmodell zur Abbildung der physischen Gegebenheiten

der Realität in der Simulation entwerfen. Die folgende Grafik verschafft einen groben Überblick über die Art der Daten:

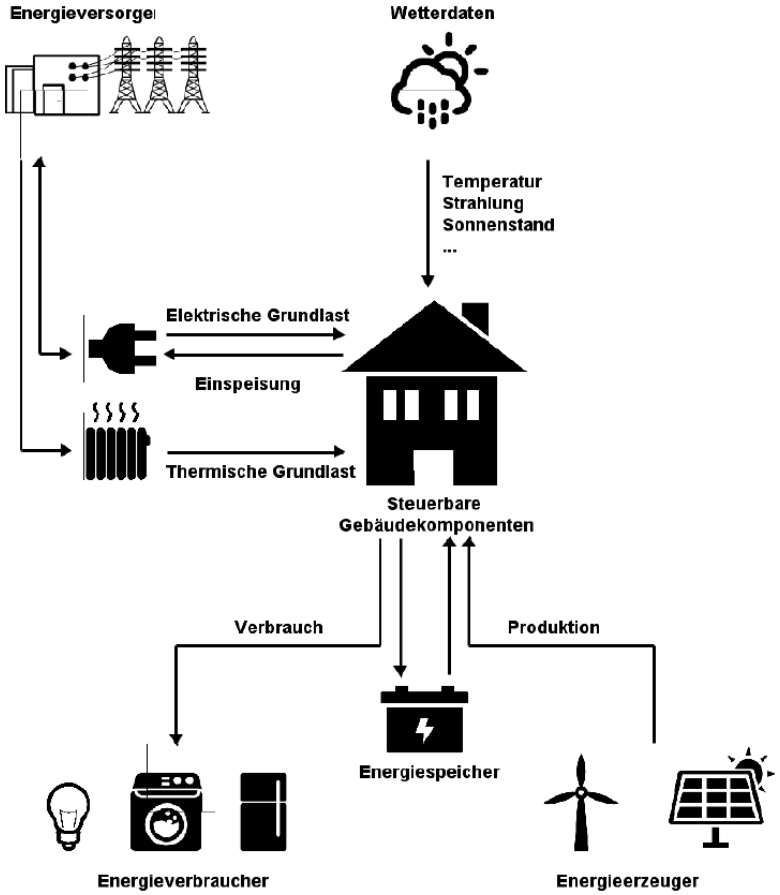


Abb. 10.3: Überblick der Daten [6]

Aus der Grafik ist abzuleiten, dass das System Daten zu den Komponenten selbst und den Energieflüssen (Verbrauch, Speicher, Produktion), den steuerbaren Gebäudekomponenten, der Grundlast sowie den Wetterdaten verarbeiten muss.

In Abschnitt 10.2.4.2 wird außerdem näher auf das Datenmodell des Systems eingegangen.

10.2.3.1 Szenarien

Eine Anforderung an die Projektgruppe war es, das Zusammenstellen bzw. Konfigurieren einer beliebigen Anzahl an Szenarien zu ermöglichen, die dann die Simulation durchlaufen. Die unterschiedlichen Rahmenbedingungen, die zuvor erläutert wurden, ermöglichen dementsprechend eine Vielzahl von möglichen Szenarien aufgrund einer großen Konfigurationsvielfalt. Szenarien beinhalten Kenndaten über das Gebäude, dessen Grundlast (in Strom- und Wärmebedarf), die nicht steuerbar ist, über Eigenschaften der steuerbaren Komponenten wie deren Dimensionierung, sowie von zwei Wetterdatenquellen, der Vorhersage und den reellen Daten. [6]

10.2.3.2 Fahrpläne

Das Energiemanagementsystem berechnet und erzeugt anhand der Simulation im 15-Minuten-Takt einen Fahrplan für die Energieversorgung. Die Anlage, ob simuliert oder real, verhält sich gemäß dieses Fahrplans. [6]

> „Ziel der Fahrplanerstellung ist die Eigenverbrauchsoptimierung zur Minimierung der Energiebezugskosten im Rahmen der definierten Simulationsdauer.“ [6]

Zudem sollte es möglich sein, die Ergebnisse der Eigenverbrauchsoptimierung evaluieren zu können. Dazu wird zum einen der Vergleich zu einem ungesteuerten bzw. nicht optimierten Fahrplan gezogen und die Frage gestellt, ob letztendlich mehr oder weniger Energie aus dem öffentlichen Netz bezogen wird. Zum anderen werden Szenarien miteinander verglichen und die Frage gestellt, welche Parameteränderungen positive Ergebnisse erzielt haben. [6]

10.2.4 Entwurf

Das Kapitel Entwurf erläutert, wie einige Entwurfsentscheidungen umgesetzt wurden, die einen Bezug zum Systemaufbau, der Kommunikation im System, dem Datenmodell und zu den Hardwarekomponenten haben.

10.2.4.1 Architekturüberblick

Die Projektgruppe erarbeitete die Architektur des Softwaresystems im Sinne einer Top-Down-Analyse und startete daher mit einer groben Sicht auf das System, beschrieb dann die Schnittstellen und anschließend die Module der Subsysteme. [6]

> „Das Energiemanagementsystem ist für die Steuerung und Optimierung des Energieverbrauchs zuständig. Das Simulationsframework stellt Daten bereit, die zur Simulation von EVS-Komponenten benötigt werden. Weiterhin wurde in den Systemanforderungen festgehalten, dass EMS und Simulationsframework konfigurierbar sein sollen. Daher wurde ein drittes System, die Steuerungsebene, hinzugefügt, das als Benutzerschnittstelle dient. Im Rahmen der Projektgruppe soll das EMS als Produkt gesehen werden. Das bedeutet, dass dieses so erstellt werden muss, dass es eigenständig, also ohne die simulierten EVS-Komponenten, funktionieren kann.“ [6]

Das EMS optimiert das Zusammenspiel von Energieerzeugern, -verbrauchern und -speichern hinsichtlich einer Zielfunktion über einen festgelegten Zeitraum. Es enthält ein Optimierungssystem und eine Auswertungskomponente und kommuniziert mit den Komponenten über Steuersignale und Zustände. Für die Optimierung bedarf es Prognosen und aktuelle Daten. Das EMS sollte so konzipiert sein, dass es in der realen Welt eingesetzt werden kann. [6]

Das SF dient zum Test des EMS. Es simuliert möglichst realitätsnah das Gebäude samt aller Eigenschaften (Gebäude, EVS-Komponenten, Umgebung/Wetter). [6]

Die SE ist die Schnittstelle zwischen Anwender und EMS/SF. Es ist zuständig für die Konfiguration, Koordination (Start/Stopp von Optimierung/Simulation) und Visualisierung. [6]

Die Projektgruppe sollte das EMS als Produkt des Projekts sehen. „Das bedeutet, dass dieses so erstellt werden muss, dass es eigenständig, also ohne die simulierten EVS-Komponenten, funktionieren kann.“ [6]

> „Neben diesen drei Systemen existiert eine zentrale Datenbank, welche Konfigurations- und Simulationsdaten speichert, [. . .] [sowie] die simulierten EVS-Komponenten.“ [6]

Die folgende Grafik zeigt das bekannte System auf diesem Stand des Entwurfs inklusive der zentralen Datenbank, Angaben über in etwa zu verarbeitende Daten und der Anbindung der simulierten EVS (Energieversorgungssystem)-Komponenten von Beckhoff:

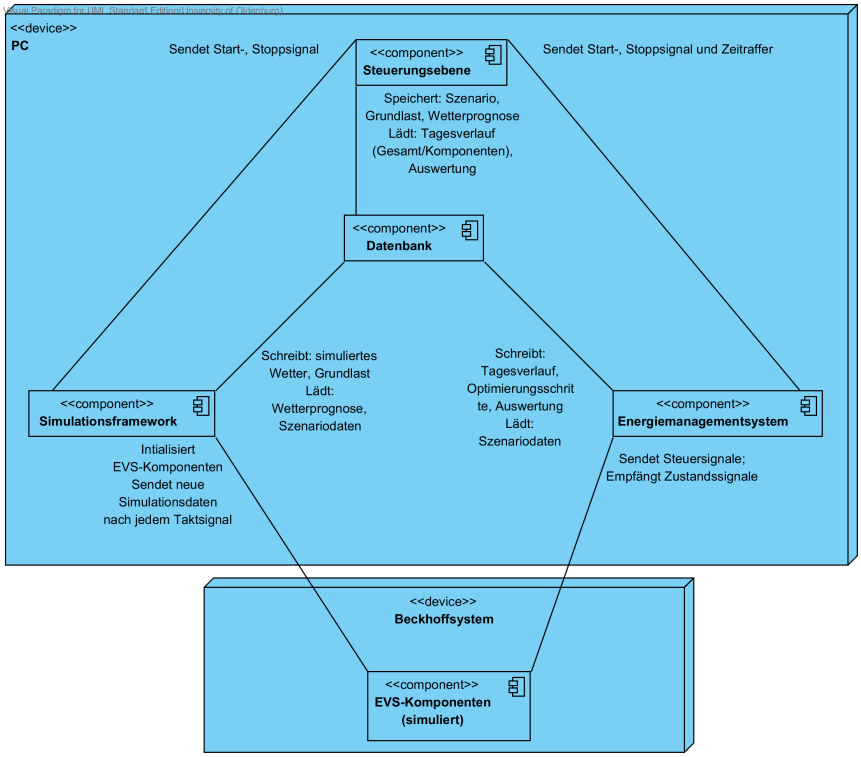


Abb. 10.4: Systemüberblick [6]

> „Infolgedessen wurden EMS und Simulationsframework als eigenständige Programme konzipiert, welche zwar miteinander kommunizieren können, allerdings bis auf kleine Ausnahmen [ClockCommunication -> ClockGenerator im EMS -> Proxy-Zeitgeber im SF] nicht aufeinander angewiesen sind. Abschließend wurde die Steuerungsebene (SE) in diese Zwei-Programm-Lösung integriert, indem sie als Standalone-GUI genutzt wird.“ [6]

Diese Ausnahmen gehören zur Taktsteuerung des Systems, denn „[. . .] das SF muss den EVS-Komponenten Simulationsdaten zu dem im EMS aktuellen Simulationsschritt liefern. Um das zu ermöglichen gibt es einen Zeitgeber (Clockgenerator) im EMS, der einen Proxy-Zeitgeber im SF synchron hält.“ Diese Kommunikation läuft über die Schnittstelle ClockCommunication. [6]

„Es wäre aber ohne viel Aufwand möglich diesen durch einen richtigen Zeitgeber zu ersetzen, falls man das Simulationsframework ohne EMS starten möchte.“ [6]

Die folgende Grafik verdeutlicht die Struktur des Systems in Verbindung mit den implementierten internen Schnittstellen:

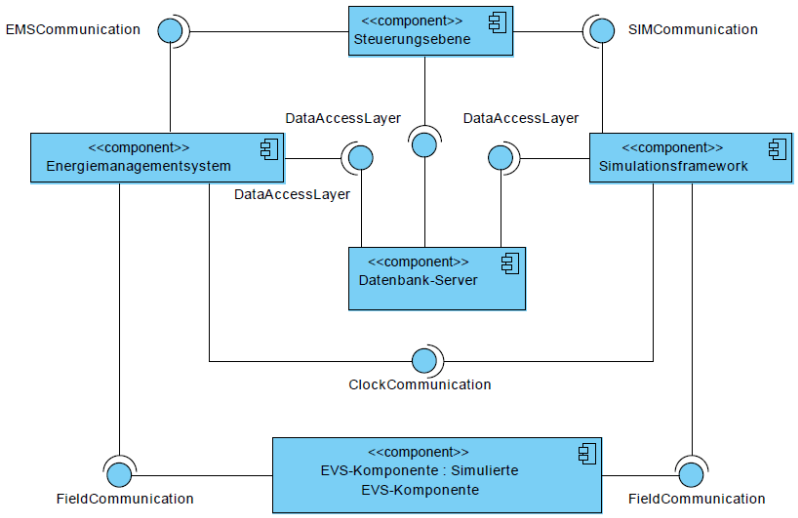


Abb. 10.5: Systeminterne Schnittstellen [6]

Die Schnittstellen EMSCommunication & SIMCommunication regeln die Kommunikation zwischen SE und EMS bzw. SF und übermitteln unter anderem Start- und Stoppsignale und Identifikatoren (IDs) ausgwählter Szenarien. [6]

Die Schnittstelle DataAccessLayer existiert mehrfach. Sie regelt zum einen die Kommunikation zwischen SE und Datenbank (DB) und übermittelt unter anderem Szenarien und Wetterdaten zur Verarbeitung bzw. Speicherung. Zum anderen regelt sie die Kommunikation zwischen EMS bzw. SF und der DB und übermittelt unter anderem Daten zum Tagesverlauf, zur Optimierung und Fahrpläne. [6]

Die Schnittstelle FieldCommunication regelt die Kommunikation zwischen EMS bzw. SF und den EVS-Komponenten und übermittelt unter anderem Steuersignale, Simulationsdaten und Werte, falls sie beispielsweise ausgelesen werden sollen. [6]

10.2.4.2 Datenmodell

Um das Datenmodell für das Softwareprojekt festlegen zu können, musste zunächst ermittelt werden, welche Daten überhaupt relevant sind. Die Projektgruppe stellte fest, dass eine exakte Repräsentation des physischen bzw. realen Umfelds nicht praktikabel sei, da dann zu viele Daten erhoben würden, deren Anwendung allerdings fraglich sei. Eine Möglichkeit war laut Projektgruppe, das Datenmodell auf die, wie in Kapitel 10.2.3 erläutert, zuvor ermittelten Daten bzw. Parameter zu reduzieren. Allerdings habe die Software dann nur wenige Konfigurationsmöglichkeiten für die Szenarien, weshalb diese Variante verworfen wurde. Stattdessen entwickelte die Projektgruppe ein abstraktes Modell der Umwelt. Dieses abstrakte Modell ist kein festes bzw. direktes Datenmodell. Dadurch werden die Erweiterbarkeit des Systems und die Konfigurationsmöglichkeiten für die variablen Szenarien gewahrt, denn so sind die EVS-Komponenten nicht mehr direkt von den Daten abhängig. Die Werte und auch der Aufbau der Datensätze werden variabel beschrieben, so können später einfach weitere EVS-Komponenten hinzugefügt werden, die während der Entwicklung der Software eventuell noch nicht berücksichtigt werden konnten. [7]

> „Diese Abstraktion führt dazu, dass es in dem Datenmodell nicht mehr klar Wetterdaten, Grundlastdaten, etc. gibt, sondern Inputdaten, denen beliebig viele Inputattribute zugeordnet sind. Den Attributen sind dabei wiederum beliebig viele Inputwerte zugeordnet. Mit diesem Modell der Inputdaten können die Datensätze beliebig erweitert und auch neue Datensätze, die bisher nicht vorgesehen waren, hinzugefügt werden ohne eine Änderung am Modell vornehmen zu müssen.“ [6]

Vorteilhaft an diesem abstrakten Datenmodell ist, dass es einfach erweiterbar und variabel ist. Nachteil daran ist, dass Daten nicht mehr eindeutig zugeordnet werden können. Wenn beispielsweise Daten über den Sonnenstand einer PV-Anlage in einem System mit einem festen Datenmodell verarbeitet werden sollen, ist für das System eindeutig, dass die Daten über den Sonnenstand der PV-Anlage zuzuordnen sind, da sie 1:1 zugeordnet sind. Das abstrakte Modell allerdings kennt dagegen nicht direkt beispielsweise PV-Anlagen, sondern nur allgemein EVS-Komponenten, denen beliebig viele Attribute zugeordnet werden, die wiederum die Parameter der Komponenten beschreiben.

> „Dieses Mapping der Parameter auf Inputdaten muss im abstrakten Modell auch wieder ermöglicht werden. Hierzu wurden sowohl die Inputattribute der Inputdaten als auch die Attribute der EVS-Komponenten mit Semantiken ausgestattet. Stimmt die Semantik eines Attributs mit der eines Inputattributs überein, wird es mit diesem beschrieben. Dabei ist jedoch zu beachten, dass die bei den Inputdaten vorhandenen Semantiken einzigartig sind, da sonst kein deterministisches Mapping möglich ist. Mit diesem abstrahierten Datenmodell ist es nun möglich das System mit allen wichtigen Informationen zu versorgen, die für die Simulation benötigt werden.“ [6]

10.2.4.3 Hardwareaufbau

„Als Hardware für die EVS-Komponenten stehen drei Embedded PC's der Firma Beckhoff zur Verfügung. [...] Untereinander sind die PC's per Ethernet und EtherCAT, einem Realzeit-Ethernet Protokoll [kurze Zykluszeiten, niedriger Jitter -> exakte Synchronisation, niedrige Hardwarekosten], verbunden. Jeder der PC's verfügt über die TwinCAT-Runtime, auf der Software-Module in Echtzeit ausgeführt werden können. Für die Kommunikation mit den EVS-Komponenten steht ein OPC Unified Architecture (OPC UA) Server zur Verfügung, der speziell die Ansteuerung der Komponenten der TwinCAT-Runtime ermöglicht." [6]

Abb. 10.6: Beckhoff EVS-Komponenten [6]

Die Embedded PCs der Firma Beckhoff führen lediglich die zu simulierenden EVS-Komponenten des Systems aus. Alle anderen Systembestandteile wurden auf einem separaten PC ausgeführt, wobei die simulierten EVS-Komponenten das Verhalten realer Komponenten nachbilden sollten. [6]

10.2.4.4 Komponentenkommunikation

Es wurde eine Schnittstelle benötigt, um über Protokolle mit den EVS-Komponenten kommunizieren zu können. Durch die Anforderungserhebung zu Beginn des Projekts existierten für die Auswahl des Protokolls die Voraussetzungen, dass simulierte sowie auch reale EVS-Komponenten angesteuert werden sollten, dass das Protokoll die ansteuerbaren EVS-Komponenten-Typen nicht einschränken solle, und dass das Protokoll möglichst verbreitet sein solle. Diese Voraussetzungen wurden dementsprechend bei der Auswahl des Protokolls berücksichtigt. Die vier Protokolle EtherCAT, ADS, IEC 61850 und OPC UA (IEC 62541) wurden infolge dessen verglichen und letztendlich OPC UA als das am besten geeignete Protokoll identifiziert, da es standardisiert sei, da es in der Industrie relativ weit verbreitet sei und da der Implementationsaufwand durch schon vorhandene Bibliotheken relativ gering sei. Zudem eignete sich OPC UA auch für die Kommunikation mit den EVS-Komponenten von Beckhoff, da Beckhoff eine entsprechende Server-Implementierung anbietet. [6] [5]

> „Entwurf der Komponentenkommunikation
> Für die Modellierung der Komponentenkommunikation gab es generell zwei Ansätze. Der eine Ansatz sieht eine direkte Verwendung der Komponenten über OPC UA vor. Hier wird die Verwaltung und Kontrolle der EVS-Komponenten den zugreifenden Systemkomponenten überlassen. Der andere Ansatz schaltet eine zentrale Steuerungskomponente (weiterhin EVS-Component-Manager genannt) zwischen die Systemkomponenten und die EVS-Komponenten. Durch die Zwischenschaltung des EVS-Component-Managers kann dieser Zugriffe auf mehrere EVS-Komponenten optimierter ausführen (zum Beispiel durch parallele Ausführung dieser Zugriffe). Die Verwendung eines EVS-Component-Managers führt auch zu einer leichteren Handhabung der Komponenten bei Ausfällen oder beim Starten anderer Simulationen, da nur noch der EVS-Component-Manager über solche informiert werden muss und dieser dann die betroffenen EVS-Komponenten korrekt beenden kann. Aufgrund dieser Vorteile basiert die entworfene Komponentenkommunikation auf der indirekten Variante über einen EVS-Component-Manager. Um eine Einschränkung des Systems auf OPC UA zu vermeiden, wurde bei der Modellierung der eigentlichen EVS-Komponenten eine Schnittstelle für solche entworfen. Mit dieser Schnittstelle können dann EVS-Komponenten erstellt werden, die verschiedene Kommunikationsprotokolle verwenden. Durch diese Schnittstelle wird offen gehalten noch weitere Protokolle zusätzlich zu OPC UA umzusetzen. Die in Anforderung EMS-80A, EMS-100A und SF-90 geforderten Funktionen sind dabei in der Schnittstelle der EVS-Komponente und des EVS-Component-Managers berücksichtigt und entsprechend in der späteren Umsetzung dieser Struktur realisiert.“ [6]

Vorteile an OPC UA sind unter anderem, dass es technologisch gesehen branchenneutral ist, auf allen Betriebssystemen und sogar ganz ohne Betriebssystem als On-Chip-Lösung läuft und in jeder Programmiersprache umsetzbar ist, wobei derzeit

Stacks in ANSI C/C++, .NET und Java verfügbar sind. Zudem hat OPC UA laut Eigenaussage der OPC Foundation die größte Verbreitung im Bereich Automatisierung. Der Zugriff auf die Richtlinien und den Code ist frei. [4]

10.2.5 Fazit der Ecob PG

In diesem Abschnitt wird zunächst das Projekt reflektiert zusammengefasst. Anschließend werden im Ausblick mögliche Verbesserungen bzw. Erweiterungen aufgezeigt.

10.2.5.1 Reflexion

Bei den anfänglichen Seminaren in der Anlaufphase des Projekts stellte sich heraus, dass sich starke Wissensmonopole in den jeweiligen Themengruppen der Seminare bildeten. Ohne dieses Wissen adäquat weiterzugeben, bilden sich zusätzliche Risiken für den reibungslosen Ablauf des Projekts, falls einzelne Personen mit viel Expertenwissen ausfallen. Zudem gingen die Themengruppen im Verlauf des Projekts „fast nahtlos“ in gleich bezeichnete Rollen über. Während des Projekts wurden die Rollen einmalig zur Hälfte der Projektlaufzeit getauscht. Dabei zeigte sich, dass ein Rollentausch sehr viel Einarbeitungszeit benötige, der nicht unterschätzt werden solle. Prinzipiell sei es jedoch sinnvoll, die Rollen regelmäßig zu tauschen, um allen Projektmitgliedern die Möglichkeit zu geben, in den verschiedenen Bereichen Erfahrungen zu sammeln. Damit sich die Arbeit jedes einzelnen Mitglieds hinsichtlich der Projektziele in die richtige Richtung entwickle, sei weiterhin intensive Kommunikation wichtig. Dazu wurde von der Projektgruppe ein vom Department zur Verfügung gestellter Projektgruppenraum genutzt. Dort wurden auch regelmäßige Treffen abgehalten. Verbindliche Treffen fanden seit Projektbeginn zweimal wöchentlich statt, sowie ab der Hälfte der Projektlaufzeit zusätzlich wöchentlich ein Standup. Unverbindliche Treffen waren zum Beispiel Programmiertreffen, wobei die Begeisterung dafür mit der Zeit abnahm. Daher sei es eventuell sinnvoller, diese Treffen ebenfalls als verbindlich einzustufen. [6]

Die Projektgruppe Ecob verwendete als Basis ihres Vorgehensmodells den Unified Process und passte ihn, soweit möglich, auf die spezifischen Anforderungen ihres Projekts an. Dabei gab es Schwierigkeiten. Unter anderem gehörte dazu die Bearbeitungszeit der Tasks, denn da dies kein Projekt in Vollzeit war, machte die Forderung der Erledigung der Tasks innerhalb eines Arbeitstages beizubehalten und täglichen Besprechungen wenig Sinn. Die Projektgruppe ersetzte diese Forderungen durch das wöchentliche Standup. Zudem hält es die Projektgruppe für sinnvoll, die Iterationsintervalle zu verkürzen, um eine Konzentration der Arbeit auf das Ende der Schritte zu reduzieren und je Iterationsschritt den Workload zu verringern. Dadurch sei ein konstanteres Arbeitsverhalten zu erwarten. Die von den Mitgliedern geleistete Arbeit wurde ab der zweiten Hälfte des Projekts zweiwöchentlich reflek-

tiert und der Fortschritt in regelmäßigen Gesprächen den Projektbetreuern berichtet. Dies sorgte für ein besseres Controlling durch das Projektmanagement und steigerte die Motivation der Projektmitglieder. Wurden im Projektplan nötige Änderungen erkannt, konnte der Plan bei Bedarf auch noch angepasst werden. Die Projektgruppe empfand, dass es vermutlich sinnvoller sei, statt der regelmäßigen Reflexion und dem Controlling die Ergebnisse nach jedem Iterationsschritt vorzustellen. Dies war aufgrund der Einschränkungen durch das gewählte Vorgehensmodell nicht ohne weiteres möglich, da zunächst einige konzeptionelle Arbeit geleistet werden musste und wichtige Bausteine erst integriert werden mussten, bevor sie überhaupt präsentationsbereit waren. Trotz des größeren Aufwands des Vorstellens der Ergebnisse nach jedem Iterationsschritt erwartet die Projektgruppe im Fazit, dass dies zu mehr Motivation und damit sorgfältigerer Arbeit geführt hätte. Ein weiterer Gesichtspunkt sei der Umstand gewesen, dass die Projektleiter, die ebenfalls von den Studierenden aus der Projektgruppe gestellt werden, kaum bzw. mangelnde Erfahrung in Teamführung haben, so dass es zu ausgeprägten Fehlschätzungen bei den Bearbeitungszeiten für Arbeitspakete gekommen war. Zudem war es schwierig vorherzusehen, welcher Teilnehmer wann wie viel Arbeitszeit einbringen kann. Dabei gilt zu berücksichtigen, dass das Projekt nicht in einem echten Unternehmen und nicht in Vollzeit durchgeführt wurde. Im Rahmen des Projekts herrschte freie Zeiteinteilung, sofern die geforderten Stunden des Iterationsschritts erreicht wurden. Phasenweise wurde die Einbringung von mehr bzw. weniger Zeit festgestellt, wenn zum Beispiel Klausurenphasen waren. Um dieser Ungenauigkeit entgegenzuwirken, wird empfohlen, die „Blocker“ zu identifizieren, die Anforderungen zu priorisieren und sonstige mögliche Einflüsse, soweit möglich, zu berücksichtigen. [6]

Die Projektgruppe stellte fest, dass die Evaluation und die Testphase der Software nahezu verknüpft waren, wobei einige Bugs behoben werden mussten. Beispielsweise wurde unter relativ hohem Aufwand festgestellt, dass Rechenfehler aufgrund unterschiedlicher Einheiten beim Datenaustausch passierten. Dies wäre durch das Festlegen und Dokumentieren der Einheiten für Interfaces zu Beginn vermeidbar gewesen. Die Projektgruppe hätte im Nachhinein auch mehr als die zwei vorgesehenen Mitglieder für diese zwei überlappenden Phasen eingesetzt. Die kontinuierliche Integration durch Jenkins sei zu empfehlen, da sich unter anderem Fehler in Latex-Dokumenten schnell aufdecken ließen. Dabei sollte jedoch bedacht werden, dass die kontinuierliche Integration einen zusätzlichen Zeitaufwand für Wartung und Betrieb beansprucht und allgemein die Akzeptanz einer Integrationsumgebung schnell sinkt, wenn der Aufgabe nicht konsequent nachgegangen wird. Dann könne sie auch zur Last werden. Doch trotz aller Dinge, die im Projekt nicht nach den Vorstellungen bzw. schlecht gelaufen sind, sei das Projekt ein Erfolg. Das Ziel sei erreicht und die Projektgruppe habe viel gelernt. [6]

10.2.5.2 Ausblick

Im Ausblick erklärt die Projektgruppe, dass es in dem Projekt in erster Linie darum gegangen sei, ein sogenanntes Proof of Concept für Energiemanagementsysteme

zu erstellen, weshalb der Fokus auf der Erstellung eines Gesamtsystems gelegen habe. Im Detail seien allerdings noch Verbesserungsmöglichkeiten aufgefallen, beispielsweise bei der Erstellung von Samples. Auch hätte man ein objektrelationales Mapping in der zentralen Datenbank verwenden können, was erst später aufgefallen sei. Außerdem sollten für die Kommunikation in öffentlichen Netzen außerhalb von Laborbedingungen Verschlüsselungsmechanismen integriert werden. [6]

10.3 Fazit: Die Projektgruppe (V)irtuelles (K)raftwerk - Gemeinsamkeiten und Unterschiede

Gemeinsamkeiten zwischen den Projektgruppen Ecob und VK sind die Fernsteuerung von Energieerzeugungsanlagen, die Ermittlung optimaler Fahrpläne unter Berücksichtigung des Eigenverbrauchs, sowie die Schnittstellen und Protokolle, die bei einem ähnlichen Aufbau des Software wiederverwendbar sein könnten. Je nachdem, welche Programmiersprache im Projekt VK verwendet wird, könnten auch Teile des Codes wiederverwendet werden.
Der Unterschied bei einem VK ist, dass die durch die Anlagen erzeugte Energie auf dem Energiemarkt verkauft bzw. gehandelt werden soll und dann im verhandelten Zeitraum zur Verfügung gestellt wird. Bei der Ermittlung optimaler Fahrpläne unterscheidet sich das VK in der Hinsicht, dass unter Berücksichtigung des Energiemarkts bzw. der Energiepreise auf dem Energiemarkt optimiert wird. Weiterhin steuert das Virtuelle Kraftwerk dieselben Typen von Energieerzeugungsanlagen wie das EMS von Ecob, jedoch auf einer Metaebene. Statt die Optimierung auf ein autonomes Gebäude zu beziehen, werden geographisch verteilte Energieerzeugungsanlagen in Kombination gesteuert. Zudem muss bei einem VK im Vergleich zum Energiemanagementsystem von Ecob ein größeres Augenmerk auf die Modellierung der Systemschnittstellen gelegt werden, da das Steuerungs- bzw. Managementsystem des VKs standortunabhängig von den dezentralen Energieerzeugungsanlagen und deren Komponenten ist, sich also nicht unbedingt in einem gemeinsamen Kommunikationsnetz befinden. Für die Modellierung eines derart intelligenten Energienetzes bzw. Smart Grids wurde in den letzten Jahren das sogenannte Smart Grid Architecture Model entwickelt. Dieses umfangreiche Entwicklungsmodell kann unter anderem bei der Modellierung der Systemschnittstellen helfen. [3]

Literaturverzeichnis

[1] Bundesministerium für Wirtschaft und Technologie. *E-Energy – IKT-basiertes Energiesystem der Zukunft*. 2014. URL: http://www.bmwi.de/DE/Mediathek/publikationen,did=630426.html.

[2] Sebastian Lehnhoff. „Vorlesung Smart Grid Management“. Vorlesung. OFFIS-Institut für Informatik, 2015.

[3] Alexander Neumann. *Angewandte Forschung: Intelligentes Stromnetz modellbasiert entwickeln*. letzter Zugriff 04.06.2015. URL: http://goo.gl/v1gsnP.

[4] OPC Foundation. *OPC Unified Architecture - Interoperabilität für Industrie 4.0 und das Internet der Dinge*. 2015. URL: https://opcfoundation.org/wp-content/uploads/2015/04/OPC-UA-Interoperability-For-Industrie4-and-IoT-DE1.pdf.

[5] OPC Foundation. *Smart Metering with OPC UA Enabled Intelligent Devices*. 2015. URL: https://opcfoundation.org/wp-content/uploads/2015/01/OPC-UA-SuccessStory-SmartMetering-RegioIT-v1.pdf.

[6] Projektgruppe Ecob. „Hardware-basierte Simulation energieautonomer Gebäude“. Projektabschlussbericht. Universität Oldenburg, Department für Informatik, Abteilung Umweltinformatik, 2014 / 2015.

[7] Thomas Röhrich. *Datenmodellierung und generische Datenmodelle*. letzter Zugriff 04.06.2015. URL: http://access-stammtisch.de/downloads/DM%20Vortrag%20thr%20Stand%20040209.pdf.

[8] Thomas Sauerbier. *Theorie und Praxis von Simulationssystemen: Eine Einführung für Ingenieure und Informatiker*. Hrsg. von Otto Mildenberger (Hrsg.) 1. Aufl. Vieweg, 1999.

[9] Wikimedia Foundation. *Stromnetz*. letzter Zugriff 04.06.2015. URL: http://de.wikipedia.org/wiki/Stromnetz.

Kapitel 11
Projektmanagement in der Projektgruppe

Immo Sanders-Sjuts

Zusammenfassung Dieses Kapitel wird einen Aufschluss darüber geben, wie Projektmanagement in einem akademischen Rahmen, mit Beteiligten, die nicht in Vollzeit zur Verfügung stehen, ausgeführt werden kann. Wichtige Punkte dieser Ausarbeitung sind die möglichen Vorgehensmodelle und ihre notwendigen Anpassungen für den Rahmen einer studentischen Projektgruppe. Im Einzelnen werden dies die (agilen) Vorgehensmodelle *Scrum*, *eXtreme Programming* und *Unified Process* sein. Eine Empfehlung zur Umsetzung des Projektmanagements zum Nutzen dieser Projektgruppe wird abschließend vorgestellt.

11.1 Einleitung

Projektmanagement gehört zu den wichtigsten Tätigkeiten während der Vorbereitung und Durchführung eines Projektes. Ohne ein klares gemeinsames Ziel und einen Weg dieses zu Erreichen versagen selbst Teams aus den besten Entwicklern, wie Jahr für Jahr im *Catalogue of Catastrophe* [2] nachzulesen ist. Insbesondere in der Softwareentwicklung sind endgültige Ziele oft unklar. Sie formulieren sich in Wünschen von Funktionalität, Aussehen und Performanz, entwickeln sich aber im Verlauf eines Projektes, durch die zunächst nicht überschaubare Komplexität weiter. Diese sich oftmals verändernden Anforderungen aufzugreifen, zu verwalten und Stück für Stück umzusetzen ist somit Aufgabe des Projektmanagements. Diesem dynamischen Prozess geschuldet ist es, dass sich aus dem klassischen Wasserfallmodell heraus, agile Verfahren entwickelten, die diesen Prozess berücksichtigen. Der Vorteil dieser Verfahren im Umgang mit sich weiter entwickelnden Anforderungen ist die kontinuierliche Iteration eines Prozesses. Damit wird erreicht, dass Anpassungen berücksichtigt werden können. weitere Vorteile beinhalten insbesondere auch die Möglichkeit, den Fortschritt im Projekt besser erkennen und einschätzen zu können.

Carl von Ossietzky Universität Oldenburg
E-mail: immo.sanders-sjuts@uni-oldenburg.de

Dies wird zum einen durch den Abschluss kleiner Teile des Gesamtproduktes und zum anderen durch den Rückblick auf diese Teilerfolge erreicht.

Weitere Tätigkeiten im Projektmanagement umfassen unter anderem das Personal- und Budgetmanagement. Diese Tätigkeiten werden im Rahmen dieser Ausarbeitung jedoch nicht weiter berücksichtigt, da ihr Einfluss auf das Vorgehensmodell gering ist. Also können sie in diesem Rahmen als unabhängig betrachtet werden.

Diese Arbeit gliedert sich in mehrere Teile. Zunächst wird ein Überblick über die Vorgehensmodelle *Unified Process*, *eXtreme Programming* und *SCRUM* diese und deren Anwendbarkeit in der Projektgruppe genauer belichten. Im Anschluss wird ein angepasstes Modell für die Projektgruppe vorgeschlagen.

11.2 Vorgehensmodelle

In diesem Teil der Ausarbeitung werden drei unterschiedliche Vorgehensmodelle vorgestellt werden. Das Augenmerk liegt dabei zusätzlich auf der Anwendbarkeit der jeweiligen Modelle innerhalb der Projektgruppe.

11.2.1 Rational Unified Process

Der Rational Uniied Process ist ein Vorgehensmodell der Softwareentwicklung, welches einen Rahmen vorgibt, innerhalb dessen eine Gruppe an Entwicklern an einem Projekt arbeiten kann. Der folgende Überblick basiert auf dem Buch von Essigkrug und Mey [3].

11.2.1.1 Best Practices

Der Unified Process basiert im Wesentlichen auf der Umsetzung der folgenden sechs erprobten Praktiken:

Iterative Softwareentwicklung Der Prozess der Softwareentwicklung wird in mehrere Durchläufe, sogenannte Iterationen, aufgeteilt. Damit soll die schon in Kapitel 11.1 angesprochene Weiterentwicklung der Anforderungen berücksichtigt werden können. Insbesondere soll es aber ermöglicht werden, kleine Teile des Produktes Schritt für Schritt fertigzustellen und auf diese Weise einen Überblick über den Gesamtfortschritt des Projektes zu erhalten.

Anforderungsmanagement Das Management der Anforderungen, also das initiale Festhalten und kontinuierliche Anpassen bestehender Anforderungen an neue Erkenntnisse. Die besondere Berücksichtigung im UP soll Sicherheit bieten, dass die Entwicklung tatsächlich in die richtige Richtung fortgeführt wird.

Kontrolliertes Einbringen von Änderungen Mittels der durch das Anforderungsmanagement festgehaltenen Änderungen sollen diese kontrolliert in das Produkt eingebracht werden. Dies geschieht im Rahmen der Iterationen durch das kontinuierliche Einarbeiten.

Komponentenbasierte Architektur Das Aufteilen des Systems in seine einzelnen Teile und die Modellierung ihres voneinander abhängigem Wirkens soll es ermöglichen strukturiert die richtige Lösung für kleinere Teilprobleme zu finden und diese robust zu bauen.

Visuelle Modellierung Die Darstellung der zu entwickelnden Modelle soll zusätzlich zur schriftlichen Modellierung auch in Grafiken und Schaubildern erfolgen. Die Visualisierung des Systems bietet damit allen Beteiligten die Möglichkeit ein gemeinsames Bild der Architektur gedanklich zu erfassen.

Kontinuierliche Qualitätsprüfung Während der Entwicklung sollen Ergebnisse ausgiebig auf Qualität geprüft werden. Dies geschieht durch Testen des erzeugten Produktes und dem Review der Implementierung. Dieses Vorgehen soll die Fehler frühzeitig erkennen und die Möglichkeit eröffnen diese im Anschluss zu beseitigen.

11.2.1.2 Struktur

Die Umsetzung der oben erläuterten sechs erprobten Praktiken geschieht auf drei Ebenen: Auf der zeitlichen Ebene wird die Arbeit iterativ strukturiert, während auf der inhaltlichen Ebene die Entwicklung anforderungsgesteuert abläuft und auf der konstruktiven Ebene die Architektur modelliert wird.

Zeitliche Ebene Die Dauer der Entwicklung wird in vier Phasen aufgeteilt, in denen unterschiedliche Schwerpunkte gesetzt werden. In diesen Phasen werden Arbeiten am Projekt mit unterschiedlicher Gewichtung zur Erreichung unterschiedlicher Meilensteine in sogenannten Iterationen ausgeführt. Die Phase, in welcher sich das Projekt jeweils befindet, gibt auch an wie weit das Projekt fortgeschritten ist.

Die erste Phase ist dient der Vorbereitung des Projektes. Als Meilenstein werden hier die zu erreichenden Ziele in Form einer Produktvision erarbeitet. Zusätzlich beginnt auch die Arbeit an den Anforderungen, welche grob erarbeitet werden.

In der zweiten Phase, der Ausarbeitungsphase, werden die Anforderungen weiter verfeinert und weitgehend festgelegt. Der Meilenstein ist die Erarbeitung eines Architekturprototypen, aus welchem das spätere System entwickelt wird. Dazu wird eine Architektur definiert und deren relevante Bestandteile innerhalb des Prototypen implementiert. Dies dient der Minimierung der gröbsten Risiken, bspw. dem Fall, das das System in dieser Form nicht realisierbar sein sollte.

Basierend auf diesem Teilergebnis wird in der dritten, der Konstruktionsphase, das System entwickelt und getestet. Am Ende wird der Meilenstein eines produktiv einsetzbaren Produktes erreicht werden.

Softwareentwicklung ist nicht mit der vorläufigen Fertiggstellung des Produktes im Labor abgeschlossen. Die Übergangsphase ist der vierte und letzte Schritt zum erfolgreichen Abschluss des Projektes. Das Produkt wird dem Kunden übergeben,

der den Umgang damit erlernt, weitere Fehler entdeckt und Daten aus möglichen Altsystemen importiert. Bei diesen Tätigkeiten wird der Kunde angeleitet und unterstützt werden, Schulungen finden statt, Fehler werden behoben und Daten konvertiert. Am Ende ist das Produkt vollständig ausgeliefert und damit der letzte Meilenstein erreicht.

Inhaltliche Ebene Die Anforderungen bestehen genauer betrachtet auch aus Anwendungsfällen, also kurzen Beschreibungen einzelner Funktionen aus Sicht von Nutzern des Systems. Anhand dieser Anwendungsfälle orientiert sich die Entwicklung. Die dadurch bestimmte Grenze des Projektes ergibt sich in den umzusetzenden Anwendungsfunktionen und die Anwendungsfälle liefern Abgrenzungen der Teile untereinander.

Konstruktive Ebene Die Architektur organisiert des Zusammenspiel aller Komponenten des zu entwickelnden Systems. Sie dient nicht nur dem Verständnis der gemeinsamen Vision aller Teilnehmer im Projekt, sondern auch der konkreten Entwicklung, indem sie klarstellt, welche Teile des Systems untereinander kommunizieren und miteinander handeln müssen.

11.2.1.3 Disziplinen

Die Unterteilung der entstehenden Arbeiten während einem Projekt hilft dabei, den notwendigen Aufwand und die zu verteilende Arbeit einzuschätzen. Unified Process beschreibt neun Disziplinen, an welchen über alle Iterationen und Phasen hinweg Arbeit durchgeführt wird. Wie schon in Abschnitt 11.2.1.2 erläutert, werden die Arbeiten mit unterschiedlicher Gewichtung in den einzelnen Phasen ausgeführt. In Grafik 11.1 aus [6] wird die zeitliche Relation der einzelnen Disziplinen zum Ausmaß ihrer jeweiligen Gewichtung über die Phasen eines Projektes hinweg deutlich.

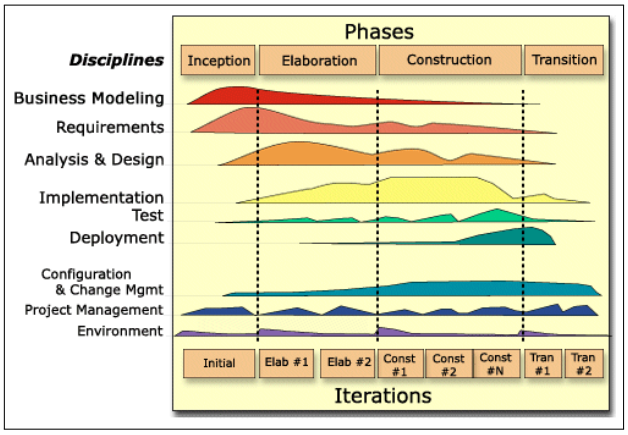


Abb. 11.1: Phasen und Disziplinen

Diese Disziplinen teilen sich in zwei unterschiedliche Kategorien. Zum einen die unterstützenden Disziplinen, die dem Team bei der Verfolgung des Projektziels Hilfe bieten und zum anderen die Disziplinen, während derer tatsächlich Arbeit am Produkt ausgeführt wird. Die unterstützenden Disziplinen sind das *Konfigurations- & Änderungsmanagement*, das *Projektmanagement* und die Bereitstellung notwendiger *Infrastruktur*. Im Folgenden werden die einzelnen Disziplinen und ihr jeweiliger Zweck kurz vorgestellt.

Geschäftprozessmodellierung Das Ziel dieser Disziplin ist es, ein besseres Verständnis über die Geschäftsprozesse zu erlangen, die das zu entwickelnde System ganz oder teilweise ersetzen, bzw. automatisieren soll. Dies geschieht über die Visualisierung der vorhandenen Geschäftsprozesse und hilft insbesondere auch die Anforderungensanalyse durchzuführen.

Anforderungen Mithilfe dieser Disziplin werden die sich weiterentwickelnden Anforderungen an das System erfasst, organisiert und dokumentiert. Diese Anforderungen stellen eine Vereinbarung aller Beteiligten über den Systemumfang dar und bieten ein grundsätzliches Verständnis für die Grenzen des Systems. Durch sie wird auch der Grundstein für die Architektur des Systems gelegt, indem einzelne Aspekte klar formuliert und für sich betrachtet definiert werden.

Analyse & Design Aus den Anforderungen heraus wird in dieser Disziplin das Design des Systems unter Berücksichtigung seiner technischen Einzelheiten erstellt. Dieses Design wird im Anschluss in eine Architektur überführt, welche stabil die bisherigen Anforderungen erfüllt, aber anpassbar genug an mögliche Änderungen bleibt. Insbesondere wird die Visualisierung der danach entstehenden Architektur mithilfe von UML durchgeführt.

Implementierung Das Ziel der Implementierung, also der Erhalt eines lauffähigen Systems, wird über die schrittweise Integration immer größer werdender Teilsysteme zu einem Ganzen verfolgt. Das System wird aufgeteilt in seine Komponenten und nach der jeweiligen Implementierung mit Tests auf Fehler geprüft und mit Hilfe von Programmcodebegutachtungen evaluiert.

Test Eine wichtige und aufwandstechnisch sehr relevante Disziplin ist das Testen. Damit ist nicht allein das Beheben der Programmierfehler in der Implementierung gemeint, sondern auch die Prüfung des Systems auf Erfüllung der definierten Anforderungen. Die Ergebnisse der Tests gehen im weiteren Verlauf der iterativen Entwicklung in die Lösung der aufgedeckten Probleme ein.

Auslieferung & Bereitstellung Die Auslieferung umschreibt drei wichtige Arten. Die interne Bereitstellung des Systems zum Testen, ebenso wie die externe Bereitstellung zum Beta-Test mit dem Kunden und die schlussendliche Auslieferung des fertigen Gesamtsystems. Hier wird ein Plan entwickelt und ausgeführt, der das System jeweils bereitstellt.

Konfigurations- & Änderungsmanagement Diese Disziplin ist für das Verwalten der Artefakte im Projekt und die Kontrolle von Änderungen an denselben zuständig. Die Unterstützung die hiermit geliefert wird soll verhindern, das konkurrierende Änderungen zu Problemen und nicht oder nur teilweise kommunizierte Änderungen zu

Missverständnissen führen. Besonders während iterativer Entwicklung ist diese Disziplin wichtig, da jeweils ein neuer Produktteil entwickelt wird, der eben versioniert werden und definitiv rekonstruierbar sein sollte. Ohne dieses Management kann es schnell zu wenig übersichtlichen Projekten kommen.

Projektmanagement Das Ziel des Projektmanagements ist von simpler Natur: Es geht darum, Kosten, Zeitpunkte und Ziele zu planen und miteinander zu verbinden. Ein besonderes Augenmerk wird im UP auf das Risikomanagement gelegt. Risiken sollen schnellstens identifiziert und eingeschätzt werden, um an Planungssicherheit zu gewinnen.

Infrastruktur Die Bereitstellung angepasster Prozesse und Werkzeuge in einer für das Projekt angepassten Entwicklungsumgebung ist das Ziel dieser Disziplin. Notwendige Richtlinien zur Vereinfachung der Kommunikation und Klarheit der entstehenden Artefakte wie bspw. Programmcode werden ebenfalls aufgestellt. All dies soll es dem Team erleichtern die Entwicklung optimal voranbringen zu können. Wichtig ist die Vermeidung von Zeitverlusten durch Probleme, die eigentlich nicht mit dem Produkt in Zusammenhang stehen.

11.2.1.4 Anwendbarkeit in der Projektgruppe

Der Unified Process ist als solches ein relativ schwergewichtiger Prozess, welcher zunächst für die Projektgruppe, wie allgemein von Essigkrug und Mey [3] vorgeschlagen, reduziert und angepasst werden müsste. Dieser Nachteil darf jedoch nicht darüber hinwegtäuschen, dass die grundsätzlichen Ideen dieses Prozessmodells durchaus einen Wert für die aufkommende Arbeit in der PG haben. Insbesondere der Fokus auf einer architekturorientierten Arbeit, welche sich an den kontinuierlich erarbeiteten Anforderungen orientiert, kann immens dazu beitragen, dass die Gruppe ein gemeinsames Ziel erkennen und dadurch auch erst verfolgen kann.

11.2.2 eXtreme Programming

Während Unified Process auf eine architekturzentrierte Entwicklung setzt, arbeitet eXtreme Programming mit einem testgetriebenen Entwicklungsprozess. Beiden ist aber gemein, dass diese Entwicklung auf den Anforderungen, bzw. speziell den Anwendungsfällen aufbaut. In eine andere Richtung als Unified Process geht eXtreme Programming allerdings auch bei der Organisation des Projektes. Im Gegensatz zu UP setzt XP stark auf die direkte und unmittelbare Kommunikation im Gespräch, statt Informationen in Dokumenten weiterzugeben.

Diese Entscheidung beruht auf der fokussierten Umsetzung fünfer Werte, die für XP im Zentrum des Entwicklungsprozesses stehen. Infolgedessen beschreibt XP insbesondere eine Menge an Techniken, die im Projektalltag eingesetzt werden. Der folgende Überblick basiert auf dem Buch von Wolf, Roock und Lippert [8] und wird diese Aspekte weiter vertiefen.

11.2.2.1 Fünf Werte

Die Erhaltung von fünf Werten bei der Arbeit des Projektes bildet die Basis für eXtreme Programming [1]. Diese Werte leiten das Team auf einer allgemeinen Ebene, sie helfen, sich angemessen zu verhalten, zu kommunizieren und mit dem Kunden gemeinsam Lösungen zu entwickeln.

Einfache Lösungen Die Entwicklung einfacher Lösungen ist einer dieser Werte: Einfache Lösungen sind schneller und kostengünstiger, dabei einfacher zu erklären, zu warten und weiterzuentwickeln. Im Zweifelsfall ist es auch simpler, komplett umzusatteln und auf eine andere Lösung aufzusteigen. Einfache Lösungen beschränken sich auf das aktuelle Problem und versuchen nicht zusätzlich gezwungen anpassbar zu bleiben. Die Logik hinter diesem Vorgehen ist, dass falls das System jemals angepasst werden müsste es simpler und zeitsparender ist, diese einfache Lösung umzustoßen und eine neue passendere zu entwickeln, als im Voraus möglicherweise auftretende Änderungen zu berücksichtigen.

Kommunikation Dieser Wert spielt eng mit dem vorherigen zusammen: Kommunikation dient zum einen dem Informationsaustausch, so dass damit auch eine Aufgabe der Dokumentation übernommen wird. Diese ist weiterhin wichtig, z.B. für spätere Teams oder den Kunden, allerdings auf diese Weise nicht mehr für den Projekterfolg entscheidend. Zum anderen ist Kommunikation auch intern für das Team wichtig. Entwickler an unterschiedlichen Teilen des Projektes sollten sich trotzdem miteinander absprechen. Gerade Meinungen aus einem anderen Teil des Systems sind häufig wichtig für den Erhalt der Synergie innerhalb des Gesamtsystems.

Feedback Damit ist die Rücksprache mit allen involvierten Parteien, über Änderungen und Weiterentwicklungen gemeint. Mögliche Fehler, im Design, ebenso wie in der Implementierung können bestens unmittelbar im persönlichen Gespräch gewonnen werden. Der Wert steht für die Sicherung der Qualität, durch Komponententests, kommunizierte Anwendermeinungen und Reviews von Programmausschnitten bzw. entwicklungstechnischen Entscheidungen. Besonderes Augenmerk liegt auf dem unmittelbaren Erhalt des Feedbacks, um den Lerneffekt möglichst optimal auszunutzen und Anpassungen unmittelbar einarbeiten zu können.

Mut spielt zusammen mit den anderen Werten, von denen XP lebt. Diese einzuhalten erfordert eine Menge davon: Einfache Lösungen, da etwas bewusst weggelassen werden muss, das die Lösung komplexer machen würde. Feedback erfordert den Mut, auch Kritik an der eigenen Lösung hinzunehmen und nicht als persönlichen Angriff zu missverstehen. Und Kommunikation erfordert den Mut, sich mögliche Missverständnisse einzugestehen, zu klären und das System möglicherweise wieder zu ändern.

Respekt Besonders die Kommunikation untereinander zeichnet sich durch den respektvollen Umgang miteinander aus. Respekt ist aber auch vor sich selbst und der Arbeit der anderen wichtig. Kein Programmierer sollte beispielsweise Programmcode liefern, der das System auseinanderbrechen lässt oder die Arbeit anderer verzögert. Wichtig ist in diesem Kontext auch das Streben nach der optimalen Lösung und damit einhergehend die Überarbeitung des eigenen Programmcodes falls nötig.

11.2.2.2 Techniken

Die Erhaltung und Kultivierung der fünf Werte wird in der Praxis durch die Anwendung von Techniken unterstützt, die dem Team helfen, seine Arbeit durchzuführen. Einen Überblick über diese Techniken gibt die Grafik 11.2. Im Folgenden wird ein Überblick über die markantesten dieser Techniken gegeben. Für eine ausführliche Beschreibung aller Techniken sei nochmal auf das Buch von Wolf, Roock und Lippert verwiesen, aus dem auch die Grafik entnommen wurde [8].

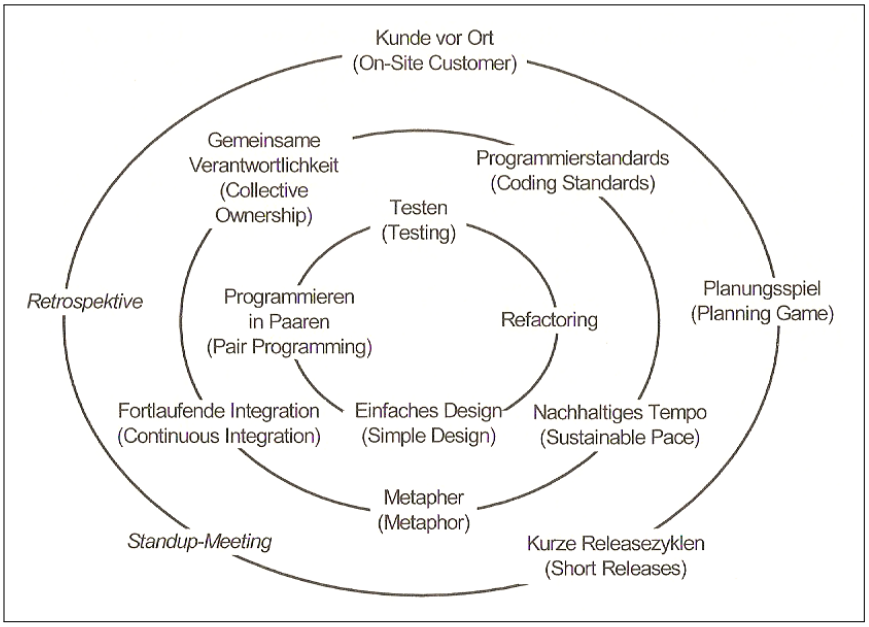


Abb. 11.2: eXtreme Programming Techniken

Kunde vor Ort Der Kunde möchte ein System entwickelt haben, das seinen Ansprüchen genügt. Als solches ist es von großem Vorteil, wenn er mit den Entwicklern über seine Ziele und Wünsche kommuniziert. Um dies möglichst einfach und schnell zu ermöglichen sollte der Kunde oder einer seiner Mitarbeiter vor Ort bei den Entwicklern anwesend sein. Wichtig ist dabei, dass er Anforderungen, die er an das System hat, mit sogenannten User-Stories beschreibt. Dies sind kleine Geschichten aus seiner Perspektive die erzählen, wie er sich den Umgang mit dem System vorstellt. Das muss auf keinen Fall formal sein, sondern kann im Gegensatz sogar Lücken aufweisen. Der Kunde nimmt an der Planung teil und priorisiert genau diese Anforderungen, damit das Team weiß, was es als nächstes entwickeln soll. Der Kunde kann durchaus seiner täglichen Arbeit nachgehen, sollte jedoch immer für Fragen zur Verfügung stehen.

Dem Kunden ist damit eine enorme Verantwortung auferlegt. Er muss die fachliche Projektplanung übernehmen, also entscheiden was wann realisiert werden soll. Dazu muss er die großen Zusammenhänge verstehen und konkrete Anforderungen formulieren. Dazu sollte ihm die Befugnis und Fähigkeit nicht fehlen, Anforderungen in Blöcke zu gliedern. Zusätzlich muss er abschätzen, wie lange die Softwareentwicklung finanziell betrachtet rentabel bleibt. Oft ist es schwierig einen qualifizierten Kunden zu haben, der all dies kann und zusätzlich noch für Fragen zur Verfügung stehen kann. Nichtsdestotrotz ist der Kontakt mit dem Kunden essentiell für den erfolgreichen Abschluss des Projektes.

Testen Mit dem Testen ist insbesondere die Durchführung der testgetriebenen objektorientierten Entwicklung gemeint. Dies bedeutet, dass die Entwickler gleichzeitig Klasse und Komponententest schreiben, wobei die tests zu einer Operation immer vor der Implementierung der Funktion geschieht. Auf diese Weise wechseln sich Programmierung und Testen schnell untereinander ab. Dies hat einige Vorteile schon während der Entwicklung.

- Es wird direkt beim Implementieren gemerkt, wenn Fehler entstanden sind. Diese können direkt behoben werden.
- Da komplexe Klassen schwieriger zu testen sind, werden Entwürfe generell von den Entwicklern einfacher gestaltet, eben um das Testen nicht unnötig zu verkomplizieren.
- Bei der Namensfindung für die Operationen ist es auf diese Weise einfacher aussagekräftigere Namen zu finden, da die Operationen beim Test schreiben aus Sicht der nutzenden Klasse betrachtet werden.

Besonders auch bei Änderungen der Struktur des Programmcodes kann mithilfe der Komponententests schnell festgestellt werden, ob dies zu unerwünschten Seiteneffekten geführt hat. Auch bei der Fehlerbehebung ist der Einsatz der Komponententests konstrutiv möglich. Der Fehler wird zunächst durch einen angepassten Tests sichtbar gemacht und nach der Behebung des Fehlers kann mithilfe eben dieses Tests sichergestellt werden, dass er nicht wieder auftritt.

Zusätzlich zu den Komponententests gehören auch Akzeptanztests zum Testen dazu. Dazu gehört die Beschreibung eines Tests aus Sicht des Anwenders. Diese Beschreibung soll anschließend automatisch ausgeführt werden. Da jedoch die Benutzeroberfläche oft nur schwierig automatisch bedient werden kann, bietet es sich an einen Akzeptanztest auch auf den alleinigen Test der Funktionalität zu beschränken. Dies ist jedoch nur bei strikter Trennung von Benutzeroberfläche und Funktionalität möglich.

Einfaches Design Wie schon zu Beginn bei den Werten von XP in Abschnitt 11.2.2.1 erwähnt, ist es grundlegend gewollt, eine möglichst simple Lösung zu wählen. Dieses Vorgehen soll zu einem inkrementellen Design führen, wobei jeweils zu dem aktuellen Problem ein optimales Design gefunden wird, welches dieses Problem möglichst einfach löst. Die schnelle Lösung führt, durch den zeitnahen Einsatz, zu einem Vorsprung beim Erkennen von Designfehlern im Vergleich zur herkömmlichen Modellierung. Dies kann und sollte dazu führen, dass das Design nocheinmal angepasst wird.

Programmieren in Paaren Die Bedeutung dieser Technik ist simpel: Zum Programmieren sitzen immer zwei Programmierer gemeinsam vor dem Rechner. Beide sind gleichberechtigt, erarbeiten gemeinsam mögliche Lösungen und implementieren sie. Selbstverständlich können nicht beide Entwickler gleichzeitig den Rechner steuern. Daher gibt es Rollen, die dynamisch jeweils nach einigen Minuten wechseln. Zum einen den Programmierer der gerade Maus und Tastatur steuert, eine Lösung kodiert und seinem Partner erklärt, was er tut und warum. Zum anderen beurteilt dieser zweite Entwickler die Lösungen des ersten und denkt über Vereinfachungen, Alternativen, etc. nach.

Diese Technik führt insbesondere (bei der notwendigen Aufmischung der Paare) zu einer Verbreitung des Wissens über eingesetzte Technologien und bestimmte Systemkomponenten im Team. Das hilft vor allem auch, wenn kurz- oder langfristig ein Teammitglied ausfällt. Dann gibt es immer noch eine zweite und später sogar mehr Personen die über dessen Bereich ausreichend Wissen haben, um die Arbeit fortzuführen. Besonders die gemeinsame Verantwortung des Teams für das Gesamtsystem profitiert von dieser Technik, da ohne das nötige Wissen nur sehr schlecht Verantwortung über einzelne Bereiche übernommen werden kann.

11.2.2.3 Anwendbarkeit in der Projektgruppe

Die Durchführung der Projektgruppe mittels eXtreme Programming würde auf einige kleinere Hindernisse stoßen. Es ist schwierig, gemeinsame Termine für die Entwicklung zu finden, wenn die Stundenplanung der Mitglieder schon fast keinen Platz für gemeinsame Planungssitzungen offenlässt. Daher kann das Utopia der gemeinsamen Entwicklung nur ein solches bleiben. Viele der Techniken fordern eine Stetigkeit und Disziplin von allen Mitgliedern, die von Studenten im Rahmen des Studiums, innerhalb eines relativ freien Frameworks, schwer geleistet werden kann.

Nichtsdestotrotz ist die Nutzung von einzelnen Techniken als sehr gut möglich zu erachten. Da die Betreuer einer Projektgruppe zumeist auch die Kunden derselben sind, kann die *Kunde vor Ort* Technik ohne Probleme und allzu große Umstellungen genutzt werden. Auch dem Programmieren in Paaren spricht organisatorisch nichts entgegen, da sich zu zweit durchaus noch Zeit für Treffen finden lässt. Gerade im Studium kann es nützlich sein die 4-Augen-Technik anzuwenden, wie der Einsatz von Arbeitsgruppen und Peer-Assessment in der Lehre zeigt. Auch gegen den Einsatz anderer Techniken spricht nichts, außer der oben erwähnten notwendigen Disziplin und Stetigkeit. Der korrekte und dauerhafte Einsatz des gesamten Vorgehensmodelles mag eine Herausforderung sein, die es nicht notwendigerweise lohnt sie anzugehen.

11.2.3 Scrum

> „Scrum is a framework, not a method or process. It is designed to be as incomplete as possible. [. . .] It has one simple command: Inspect and Adapt.“
>
> (Ron Jeffries, The Case of Scrum Dying in a Fire [5])

Scrum bietet im Vergleich zu den bisher vorgestellten Vorgehensmodellen eine andere Strategie: Es bietet ein solides, Framework bei der Durchführung eines Projektes, ohne den Freiraum des Teams unnötig zu beschränken. Dabei gibt es absichtlich ein unvollständiges Gerüst an, um jedem Team die Möglichkeit zu geben, für sich selbst den Prozess anzupassen. Ron Jeffries hat dies in seinem eingängigen Zitat auf die Spitze getrieben dargestellt.

Beschreiben tut Scrum jedoch sehr wohl ein iteratives Vorgehensmodell. Dies soll jedoch lediglich die Basis für das Team darstellen von der aus es sich weiterentwickeln kann. Eine Beschreibung dieser Basis ist im folgenden basierend auf dem Buch von Gloger [4] zu finden.

11.2.3.1 Team

Zu Beginn steht des Team von Entwicklern. Dieses Team hat jetzt den Auftrag eines Kunden erhalten und sich dazu entschlossen dieses Produkt für ihn zu entwickeln. Zunächst gehört dazu die Entwicklung einer gemeinsamen Vision. Die Vision ist wichtig über den gesamten Verlauf der Entwicklung hinweg. Sie ist der Angelpunkt des Projektes. In der Vision wird aufgezeigt, was das Produkt sowohl für den Kunden als auch für das Team bedeutet. Einigkeit über diese Vorstellung ist wichtig, sonst gibt es später im Projekt Diskrepanzen, welche den Fortschritt blockieren oder zumindest ausbremsen werden.

Zum Team gehören drei Rollen: Der Scrum Master, der Product Owner und die Entwickler. Jede dieser Rollen hat ihren eigenen Verantwortungsbereich. Dies ist ein freiwilliger Prozess, es geht hier um die Eigeninitiative, den Wunsch, die Verantwortung zu übernehmen. Insbesondere sind diese Rollen keine Positionen die übergeben werden können, sondern Personen üben die Rolle für das Team aus.

Scrum Master Der Scrum Master schützt das Team. Er leitet es an und implementiert den Scrum Prozess. Dabei ist seine Aufgabe einer unterstützenden Art. Er hilft dem Team den Scrum Prozess zu lernen und kontinuierlich zu verbessern. Dem Team werden sich immer wieder neue Hindernisse in den Weg stellen. Der Scrum Master ist bei der Beseitigung derselben und der Hilfe zum Erreichen der vom Team selbstgesteckten Ziele tätig. Er arbeitet zusammen mit allen Mitgliedern des Teams und hilft insbesondere immer dort, wo es brennt.

Product Owner Der Product Owner bringt das Projekt voran, er zeigt die klare Vision des Produktes auf und ist verantwortlich dafür, dass das Team an den gerade sinnvollsten Aufgaben arbeitet. Dafür priorisiert er die anliegenden Aufgaben für das Team und setzt sich für dasselbe während dem Entwicklungsprozess ein.

Team Das Team besteht aus den einzelnen Entwicklern. Jedes Mitglied übernimmt die Erledigung der anstehenden Aufgaben und die Verpflichtung diese zu erfüllen. Dem Team ist dabei offen, wieviel es an anstehenden Aufgaben bearbeiten kann und wird, bevor erneut Aufgaben ausgewählt werden.

11.2.3.2 Sprint

Eingangs wurde erwähnt, dass die Entwicklung in Scrum iterativ geschieht. Die Vorteile iterativer Entwicklung sind bereits oben in Abschnitt 11.2.1.1 aufgezeigt worden.

Die Grafik 11.3 zeigt die folgende Erläuterung des Scrum Prozesses visuell auf. Die iterative Entwicklung in Scrum geschieht in sogenannten Sprints, während derer das Entwicklerteam jeweils einen kleinen Teil des Produktes fertigstellt. Dieser kleine Teil heißt das Product Increment und die einzelnen Teammitglieder übernehmen zu Beginn des Sprints die Verpflichtung dieses Product Increment am Ende des Sprints auszuliefern. Sprints sind zwischen zwei und vier Wochen lang.

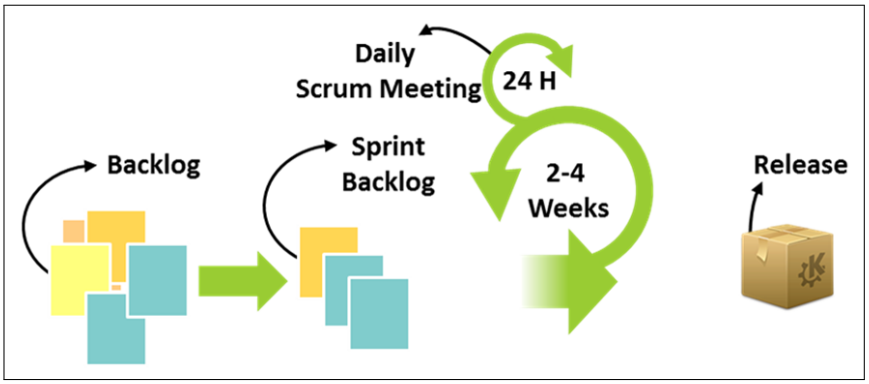


Abb. 11.3: Vorgehensmodell nach Scrum

Sprint Planning Zu Beginn trifft sich das gesamte Team zu einem Sprint Planning Meeting, wo der Umfang und Inhalt des Sprints ausgehandelt wird. Dies wird im Sprint Backlog festgehalten, der Sammlung aller Aufgaben, die bis zum Ende des Sprints von den Entwicklern bearbeitet werden. Insbesondere eine Beschreibung des Product Increments ist Ergebnis der Sitzung. Der Product Owner hat zu diesem Zeitpunkt eine aktualisierte und priorisierte Sammlung der Aufgaben zur Fertigstellung des Produktes vorbereitet, damit die Auswahl auf dieser Basis informiert geschehen kann. Diese Sammlung heißt Product Backlog und beinhaltet alle notwendigen Aufgaben zum Abschluss des Projektes. Die für einen späteren Teil des Projektes vorausgesehenen Aufgaben sind weniger detailliert und deswegen ungenauer.

User Stories Eine Variante des Scrum wie sie von Wirdemann in seinem Buch [7] beschrieben wird, benutzt das Konzept der User Stories, um mit ihrer Hilfe den Product Backlog darzustellen. User Stories wurden auch oben bei der Erläuterung des eXtreme Programming in Abschnitt 11.2 erklärt: Sie sind kurze Geschichten aus Sicht des Anwenders bei der Nutzung einer Funktionalität. Diese Geschichten lassen sich auch Anwendungsfällen zuordnen und beschreiben insgesamt das System, wie es in funktional wirken soll. Damit sind sie ideale Elemente, bzw. in ihrer Umsetzung dann Aufgaben für das Product Backlog.

Agiles Schätzen Der gerade beschriebene Einsatz der User Stories im Product Backlog bietet die unter anderem die Möglichkeit genauer agil zu planen. Mithilfe einer relativen Schätzung der Größe der User Story kann insgesamt der Aufwand innerhalb eines Sprints abgeschätzt werden. Diese Schätzung dient der Verbesserung der Projektplanung und der reduktion unnötiger Risiken. Unter Bezugnahme auf die Größenschätzungen früherer Sprints kann das Team erkennen wieviel Arbeit es innerhalb eines Sprints übernehmen kann. In diesem Zusammenhang steht dann die Zahl, die die Bearbeitungsgeschwindigkeit des Teams, also die summierte Menge der Schätzungen aller abgeschlossenen Aufgaben, ausmacht, die sogenannte Velocity. Nicht fertiggestellte Aufgaben werden mit null berücksichtigt, da nicht eingeschätzt werden kann, wieviel Arbeit noch zu erledigen war.

Mehrere Möglichkeiten eine Schätzung zu erhalten bieten sich an. Hier soll beispielshaft das Planning Poker erklärt werden. Bei dieser Technik trifft sich das Team, um gemeinsam die relative Größe der Elemente im Product Backlog zu schätzen. Dazu wird initial ein beliebiges Element geschätzt. Dieses Element muss nicht im tatsächlichen Projekt existieren, sollte aber jedem Teilnehmer bekannt und klar sein. Die Teilnehmer erhalten zum Schätzen Karten, auf denen steigende Zahlen für steigende Größe abgebildet sind. Diese Zahlen sollten dem Muster entsprechen, das geringer Aufwand in kleinen Abständen und großer Aufwand in immer größeren Abständen geschätzt werden soll. Dies ist dem Fakt geschuldet, dass große Aufgaben unübersichtlich und damit schwer einschätzbar sind. Jeder Teilnehmer erhält eine Kopie dieser Zahlenreihe. In Runden wird jetzt jeder Teilnehmer eine Zahl wählen und diese Schätzung offenbaren. Jene unter den Teammitgliedern, deren Schätzung am unteren und oberen Rand liegen, begründen ihre Entscheidung kurz, dann wird eine neue Runde gestartet, bis idealerweise alle Teilnehmer dasselbe schätzen. Der Grund für dieses Vorgehen liegt in der Berücksichtigung aller Faktoren die eine Schätzung beeinflussen. Gründe etwas als kleine Aufgabe zu betrachten sind ebenso wichtig zu erörtern wie Gründe die eine Aufgabe eventuell vergrößern. Auf diese Weise werden beide Seiten gehört und eine begründete Schätzung kann gefunden werden.

Mit dieser Schätzung kann insbesondere dem Kunden gegenüber klargemacht werden, wieviele Aufgaben das Team übernehmen kann und wie das Team zu dieser Einschätzung kommt.

Sprintalltag Innerhalb des Sprints übernehmen die Entwickler ihre freigewählten Aufgaben und bearbeiten diese. Jeden Tag zu einem festen Termin trifft sich das gesamte Team zu einer kurzen Besprechung, dem sogenannten Daily Scrum Meeting, wo jeder Entwickler berichtet, was er oder sie gestern geschafft hat, wo es Proble-

me gibt und was heute von ihm oder ihr angegangen wird. Dies geschieht in einem Stand-up Meeting, d.h. jeder Teilnehmer steht für die Dauer der Besprechung. Damit soll die Dauer dieser Besprechung natürlich begrenzt und auf die notwendige Kommunikation reduziert werden. Im Anschluss bilden sich häufig kleinere Gruppen, um spezielle Probleme zu lösen.

Sprint Review Zum Abschluss des Sprints trifft sich das gesamte Team mit allen Beteiligten, um sein Product Increment vorzustellen. Dieses Treffen heißt Sprint Review und hilft zu kommunizieren, was geschafft wurde, und wo eventuell Probleme umgangen werden mussten, aber auch welche Erfolge das Team vorzuweisen hat. Dies ist auch ein guter Zeitpunkt, um zu erkennen, inwieweit das Team sich auf der Spur gehalten hat, bzw. wo es von den Vorstellungen des Kunden abgewichen ist.

Sprint Retrospective Im Anschluss an das Sprint Review trifft sich das Team intern, um zu besprechen, wie das Vorgehen selbst zu bewerten ist. Dies ist der Punkt, an dem das Eingangszitat wieder sehr relevant wird, d.h. wo das Team sich selbst inspiziert und Anpassungen machen kann, um sich zu verbessern.

11.2.3.3 Definition of Done

Der letzte, aber in jedem Projekt sehr relevante Punkt ist der Konsens des Teams, wann eine Aufgabe als fertig zu bewerten ist. Da ein Scrum Team selbst entscheidet, wann eine Aufgabe fertiggestellt ist, ist dies häufig einer der Punkte, die öfters in der Retrospektive angesprochen werden und wo sich das Team selbst anspornt noch besser zu werden. Oftmals entscheiden sich Teams ihre eigene Definition noch weiter zu verfeinern oder zu verstärken. Beispielhaft beinhaltet dies zusätzlich zum Abschluss einer Implementierungsaufgabe das Erstellen von Tests oder das Schreiben von Dokumentation.

11.2.3.4 Anwendbarkeit in der Projektgruppe

Scrum ist als Vorgehensmodell ein sehr simples Gerüst, dass nach den Wünschen des Teams nach Bedarf angepasst werden kann. Der damit einhergehende Nachteil ist die zunächst relative Unsicherheit besonders zu Beginn eines Projektes. Der Autor hat Scrum bereits einmal erfolgreich in einem studentischen Projekt eingesetzt und diese Erfahrungen gemacht. Selbstverständlich kann Scrum nicht in dieser Form direkt als Vorgehensmodell im Studium implementiert werden, da Studenten sich nicht jeden Tag zu einem Daily Scrum Meeting und Arbeit Zeit haben, allerdings konnte der Rahmen in diesem Fall etwas gelockert auf zweimal wöchentlich angepasst werden und führte zu einem soliden Vorgehensmodell. Besonders die Anpassungsfähigkeit macht dieses Vorgehensmodell zu einem guten Ausgangspunkt für Projekte auch im Rahmen eines Studiums.

11.3 Vorschlag eines Vorgehensmodells zur Projektgruppe

Bei Betrachtung der Anwendbarkeit der einzelnen vorgestellten Modelle ergibt sich ein durchweg eher positives Bild mit der Möglichkeit auf viele verschiedene Interpretationen. Die strukturierte Herangehensweise die Unified Process über den gesamten Entwicklungsprozess betreibt hat den Vorteil einer sehr guten Planbarkeit und Fortschrittsübersicht. Die Techniken des eXtreme Programming zeigen überzeugende Vorteile auf. Und die Anpassbarkeit des Scrum Prozesses mit seiner simplen Gestaltung bietet einen leichten Einstieg.

Der Schluss diese Vorteile zu kombinieren liegt nahe. Als angedachter Vorschlag ergibt sich die Durchführung von Scrum innerhalb eines an Unified Process angelehnten Phasenmodells zur stärkeren Strukturierung des Vorgehens insbesondere auch zum Projektstart, da Scrum hierzu wenig Hilfe bietet. Dazu sollte hier, um die Planungssicherheit zu erhöhen, Agiles Schätzen mithilfe der User Stories geschehen. Im weiteren Verlauf des Projektes können dann selektierte Techniken aus dem eXtreme Programming zur weiteren Verbesserung dieses Prozesses hinzugezogen werden. So wäre insbesondere die Pair Programming Technik eine durchaus denkbare, nützliche Einführung. Dabei ist allerdings zu beachten, das eXtreme Programming auch vor Schaden warnt, der bei einer nur teilweisen Ausführung weniger Techniken entstehen könnte. Daher muss hier besonders aufgepasst werden, wie die Techniken für das Vorgehensmodell ausgewählt und angepasst werden.

11.4 Fazit

Das Feld des Projektmanagement ist weit und innerhalb dieser Ausarbeitung konnten nicht alle aktuellen Vorgehensmodelle vorgestellt werden (so fehlt z.B. Kanban). Klar geworden ist jedoch, dass die Vorgehensmodelle nicht schwarz und weiß zu bewerten sind, sondern jeweils für jedes Projekt zu evaluieren ist, welches Modell das geeignetste sein könnte. Insbesondere vor dem Hintergrund des Studiums ist Projektmanagement jedoch definitiv auch mithilfe eines angepassten agilen Vorgehensmodells möglich und hilft das Projekt zu einem gelungenen Abschluss zu leiten.

Besonders persönliche Erfahrungen werden allerdings immer eine große Rolle bei der Akzeptanz und erfolgreichen Ausübung jedes der Modelle spielen. Sicherlich sind in der Vergangenheit nicht alle Projekte, die das Wasserfallmodell genutzt haben, gescheitert und sicherlich sind auch nicht alle Projekte heutzutage erfolgreich, nur weil sie eines der neueren iterativen Modelle einsetzen. Man kann jedoch zuversichtlich sein, dass mit der wachsenden Erfahrung aller, die Erfolgswahrscheinlichkeiten steigen werden.

Literaturverzeichnis

[1] Kent Beck. *Extreme programming explained : embrace change*. 2. Addison-Wesley, 2006.

[2] *Catalogue of Catastrophe*. English. International Project Leadership Academy. 2015. URL: http://calleam.com/WTPF/?page_id=3.

[3] Andreas Essigkrug und Thomas Mey. *Rational Unified Process kompakt*. Spektrum Akademischer Verlag, 2003.

[4] Boris Gloger. *Scrum: Produkte zuverlässig und schnell entwickeln*. 4. Carl Hanser Verlag, 2013.

[5] Ron Jeffries. *The Case for Scrum Dying in a Fire*. English. März 2015. URL: http://ronjeffries.com/articles/2015-02-23-die-in-a-fire/.

[6] Eric Poinsignon. „Rational Unify Process (RUP), a Software Process Knowledge Base“. In: *CERN Computer Newsletters* Vol. XXXVII.issue no 1 (2002). URL: http://ref.web.cern.ch/ref/CERN/CNL/2002/001/.

[7] Ralf Wirdemann. *SCRUM mit User Stories*. 2. Carl Hanser Verlag, 2011.

[8] Henning Wolf, Stefan Roock und Martin Lippert. *eXtreme Programming - Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis*. 2. dpunkt.verlag, 2005.

Kapitel 12
Requirements-Engineering

Almuth Meier

Zusammenfassung In vielen Softwareentwicklungsprojekten sind eine ungenügende Anforderungsdokumentation und nachträgliche Anforderungsänderungen die Ursachen für das verspätete Ausliefern des Produktes oder sogar für das Scheitern des Projektes. Das Requirements-Engineering ist in Softwareentwicklungsprojekten die Disziplin, die für eine strukturierte Anforderungserhebung und -verwaltung zuständig und somit grundlegend für den Erfolg des Projektes verantwortlich ist. In der vorliegenden, im Rahmen der Seminarphase der Projektgruppe „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“ entstandenen Arbeit wird der Frage nachgegangen, wie in der Projektgruppe ein zweckerfüllendes Requirements-Engineering ausgestaltet werden kann.

12.1 Notwendigkeit des Requirements-Engineering

Die heutige Softwareentwicklung unterliegt immer größeren Herausforderungen [4]: Komplexere Systeme sollen mit sehr guter Qualität in kürzeren Entwicklungszeiten und mit weniger Kostenaufwand entwickelt werden. Trotzdem überschreiten dem „CHAOS-Report 2012“ der Standish Group [8] zufolge 43% der Softwareentwicklungsprojekte ihren Zeit- und Kostenrahmen und stellen dann teilweise sogar nicht die gesamte gewünschte Funktionalität bereit. Darüber hinaus scheitern 18% der Projekte, sodass sie entweder vorzeitig abgebrochen oder die entwickelten Produkte nicht verwendet wurden. Aus diesem Ergebnis der Studie und der Erkenntnis, dass es in der Praxis unmöglich ist, 100% der gewünschten Funktionalität umzusetzen, zieht die Standish Group das Fazit, dass eine höhere Kundenzufriedenheit erreicht werden kann, wenn die Entwicklung des Systems auf die 20% der Funktionalität konzentriert wird, die 80% des Systemwertes ausmacht.

Carl von Ossietzky Universität Oldenburg
E-mail: almuth.meier@uni-oldenburg.de

Um diesen Teil der Funktionalität schon zu Beginn des Projektes zu kennen, ist in Softwareentwicklungsprojekten ein gutes Requirements-Engineering notwendig. Dessen Aufgabe ist es, zu Beginn des Projektes die wesentlichen Anforderungen aller Stakeholder an das System zu erheben, das heißt zu sammeln und in geeigneter Form in einem Anforderungsdokument zu dokumentieren. Nach [6] ist es nur auf diese Weise möglich, ein von den Stakeholdern akzeptiertes System zu entwickeln. Außerdem vertritt [6] die Meinung, dass, je früher im Entwicklungsprozess Anforderungen bekannt sind, desto eher eine geeignete Lösung für sie entwickelt und konfliktäre Anforderungen erkannt und gelöst werden können. In der Praxis lässt sich nach [6] eine erfolgreiche Entwicklung durchführen, wenn zu Beginn des Projektes die wesentlichen Anforderungen der Stakeholder und die Grenzen des zu entwickelnden Systems zu seiner Umgebung festgelegt werden. Diese groben Anforderungen sind für den weiteren Entwicklungsverlauf trotz unvermeidlicher Anforderungsänderungen stabil genug, um während der Entwicklung verfeinert schließlich und umgesetzt zu werden.

Da sich Änderungen von Anforderungen, wie beispielsweise die Aufnahme neuer oder die Modifikation bekannter Anforderungen, im Verlauf eines Softwareentwicklungsprojektes nicht vermeiden lassen, ergibt sich für das Requirements-Engineering neben der Erhebung der Anforderungen eine weitere Aufgabe: die Verwaltung der Anforderungen im Laufe des Projektes und die konsistente Integration von Anforderungsänderungen in das Anforderungsdokument. Diese zweite Aufgabe wird auch als „Requirements-Management “ bezeichnet.

Um ein gutes Requirements-Engineering in einem Softwareentwicklungsprojekt zu etablieren, bedarf es eines festgelegten Prozesses und definierter Zuständigkeiten der Projektbeteiligten. In der Literatur werden verschiedene Aktivitäten, Techniken und Werkzeuge vorgeschlagen, die das Requirements-Engineering unterstützen können. Wie das Requirements-Engineering in einem konkreten Projekt ausgestaltet wird, muss für jedes Projekt spezifisch festgelegt und getestet werden. Die vorliegende Arbeit gibt einen Überblick über mögliche Aktivitäten und Techniken und trifft eine Empfehlung für eine sinnvolle Ausgestaltung des Requirements-Engineering in der Projektgruppe (PG) „Dynamische Portfoliooptimierung eines Virtuellen Kraftwerks“.

In der vorliegenden Arbeit wird zu Beginn eine grundlegende Einführung in die Thematik des Requirements-Engineering gegeben (Abschnitt 12.2). Abschnitt 12.3 enthält einen Überblick über verschiedene Aktivitäten des Requirements-Engineering, die bei der strukturierten Erhebung und Verwaltung von Anforderungen hilfreich sind, sowie Techniken, die diese Aktivitäten unterstützen. In Abschnitt 12.4 wird betrachtet, wie das Requirements-Engineering in unterschiedliche Vorgehensmodelle eingebettet ist. Abschließend wird in Abschnitt 12.5 eine Empfehlung gegeben, wie in der PG das Requirements-Engineering zweckdienlich umgesetzt werden kann, bevor die wesentlichen Ergebnisse der Arbeit in Abschnitt 12.6 zusammengefasst werden.

Wichtige Begriffe aus dem Bereich des Requirements-Engineering werden bei ihrem ersten Auftreten zur besseren Lesbarkeit *kursiv* geschrieben.

12.2 Grundlagen des Requirements-Engineering

Im obigen Abschnitt wurden die Begriffe *Requirements-Engineering* und *Anforderung* zwar bereits verwendet, trotzdem sollen sie an dieser Stelle definiert werden, da sie in der Literatur von unterschiedlichen Autoren verschieden interpretiert werden.

Das Requirements-Engineering wird von Sommerville in [7, S. 83] definiert als

> „the process of finding out, analyzing, documenting and checking“ the requirements.

Sommerville zählt alle Aktivitäten zum Requirements-Engineering, die zur Erstellung eines qualitativ hochwertigen *Anforderungsdokuments* führen, welches das zu entwickelnde System spezifiziert. In der vorliegenden Arbeit wird davon ausgegangen, dass das Requirements-Engineering zusätzlich das *Requirements-Management* umfasst, welches für die konsistente Verwaltung des Anforderungsdokuments während des gesamten Softwareentwicklungsprozesses zuständig ist. Damit wird unter anderem der Definition von [5] gefolgt.

Unter einer Anforderung soll in der vorliegenden Arbeit

> „eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“

verstanden werden [5, S. 14]. Diese Definition geht dabei deutlich über das klassische Verständis von Anforderungen hinaus, das sich lediglich auf die vom System bereitzustellende Funktionalität bezieht. Auf Grundlage dieser Definition ist es auch möglich, Anforderungen an den Entwicklungsprozess zu stellen, um beispielsweise Qualitätszertifikate zu erlangen.

In den folgenden Abschnitten wird darauf eingegangen, in welche Typen Anforderungen eingeteilt werden können, welche Kriterien es gibt, um die Qualität der Anforderungen und des Anforderungsdokuments sicherzustellen und wie ein Anforderungsdokument üblicherweise aufgebaut ist.

12.2.1 Einteilung von Anforderungen

In großen Softwareentwicklungsprojekten entsteht eine Vielzahl von Anforderungen an das zu entwickelnde System. Um alle Anforderungen systematisch in das Anforderungsdokument einpflegen zu können, ist es hilfreich, Anforderungen anhand von Kriterien in verschiedene Typen einzuteilen. Auf diese Weise ist es den Lesern des Anforderungsdokuments möglich, ganz gezielt nur bestimmte Anforderungen zu lesen. Drei gängige Einteilungskriterien werden in diesem Abschnitt vorgestellt.

Art

Anforderungen können in *funktionale* und *nicht-funktionale* Anforderungen eingeteilt werden. Funktionale Anforderungen beschreiben, welche Funktionen das Sys-

tem bereitstellen soll [5], [7]. Nicht-funktionale Anforderungen werden oft verwendet, um zu definieren in welcher Qualität das System diese Funktionen bereitstellen soll [10]. Sie müssen sich allerdings nicht immer direkt auf einzelne Funktionen des Systems beziehen, sondern können auch für das System insgesamt oder den Entwicklungsprozess definiert werden [7], [10],[5]. Sie beschreiben beispielsweise, wie zuverlässig oder wie wartbar das System sein soll, und können auch an die Benutzungsschnittstelle des Systems gerichtet sein. Zu den nicht-funktionalen Anforderungen gehören auch technologische Anforderungen, die beispielsweise zu verwendende Schnittstellenformate oder zu realisierende Soft- und Hardwarekomponenten definieren [5]. Nicht-funktionale Anforderungen, die sich auf den Entwicklungsprozess beziehen, definieren beispielsweise, wie bestimmte Tätigkeiten ausgeführt werden sollen.

Die Grenze zwischen funktionalen und nicht-funktionalen Anforderungen kann teilweise nicht scharf gezogen werden. Dies bereitet zwar bei der Einordnung der Anforderungen in das Anforderungsdokument Schwierigkeiten, für die spätere Umsetzung ist jedoch ausschlaggebend, ob die Anforderungen überhaupt im Anforderungsdokument erfasst wurden [2].

Rechtliche Verbindlichkeit

Anforderungen können ebenfalls danach eingeteilt werden, wie wichtig ihre Umsetzung für den Kunden ist. Hierbei gibt es vier Anforderungstypen, die bei der Formulierung der Anforderungen jeweils durch ein eigenes Schlüsselwort gekennzeichnet sind. Diese sind im Folgenden, neben den Bezeichnungen der Anforderungstypen, kursiv geschrieben.

Pflichtanforderungen müssen vom System umgesetzt werden, ansonsten kann die Abnahme des Systems vom Kunden abgelehnt werden. *Wunschanforderungen sollten* vom System umgesetzt werden. Sie sind für den Kunden nützlich, jedoch nicht essenziell. *Absichtsanforderungen* betreffen die zukünftige Weiterentwicklung des Systems. Das System *wird* sie zukünftig verpflichtend umsetzen. Die Berücksichtigung solcher Anforderungen in der initialen Anforderungsdokumentation hilft dabei, die Architektur des Systems so zu entwerfen, dass zukünftig angedachte Änderungen ohne großen Aufwand vorgenommen werden können. *Vorschlagsanforderungen können* vom System umgesetzt werden. Es ist sinnvoll, solche Anforderungen zu dokumentieren, um Lösungsvorschläge von Kunden zu berücksichtigen, auch wenn diese meist nicht realisiert werden [5], [11].

Detaillierungsgrad

Anforderungen können unterschiedlich detailliert formuliert werden. Zu Beginn der Anforderungserhebung werden die vom System zu erfüllenden Ziele zunächst grob beschrieben, bevor sie nach und nach in immer detailliertere Anforderungen überführt werden. [5, S. 44] unterscheidet die folgenden Detaillierungsgrade:

0. grobe Systembeschreibung: Definition der Problemstellung und der vom System zu erreichenden Ziele
1. Funktionsbeschreibung: Definition der Anwendungsfälle des Systems
2. Anwenderanforderungen: Anforderungen an das System, die sich beim Umgang eines Anwenders mit dem System ergeben ([5] teilt diese Anforderungen in zwei Grade ein)
3. Komponentenanforderungen: Anforderungen an konkrete Hard- und Softwarekomponenten des Systems

Mit Hilfe dieser Detaillierungsgrade können Anforderungen spezifisch für Stakeholdergruppen formuliert werden: Während beispielsweise für den Projektleiter eine grobe System- und Funktionsbeschreibung ausreichend ist, benötigen die Entwickler auch die konkreten Komponentenanforderungen, um das System realisieren zu können. Die Detaillierungsgrade können in einem gut aufgebauten Anforderungsdokument anhand der Gliederungsstruktur wiedergefunden werden. Ein Vorschlag für einen sinnvollen Aufbau des Anforderungsdokuments wird im nächsten Abschnitt beschrieben.

12.2.2 Aufbau des Anforderungsdokuments

Die Gliederung des Anforderungsdokuments unterscheidet sich von Projekt zu Projekt. Trotzdem muss die Gliederungsstruktur des Dokuments nicht in ihrer Gesamtheit für jedes Projekt neu entwickelt werden, denn in der Literatur existieren verschiedene Vorschläge für allgemeingültige Gliederungsstrukturen. Im Folgenden wird exemplarisch die Referenzgliederung nach dem Standard IEEE 830-1998 vorgestellt, die in Abbildung 12.1 zu sehen ist.

Die Struktur des Anforderungsdokuments ist so aufgebaut, dass im ersten Kapitel „Introduction“ neben Zweck und Aufbau des Dokuments die generelle Zielsetzung des Systems beschrieben wird. Dies entspricht Anforderungen des Detaillierungsgrads 0 (siehe Abschnitt 12.2.1). Zusätzlich sollten wichtige Begriffe entweder an dieser Stelle oder im Anhang als Glossar definiert werden, um ein eindeutiges Verständnis aller Leser bezüglich dieser Begriffe herzustellen. Im zweiten Kapitel „General description“ werden die Einbettung des Systems in seine Umgebung und die Anwendungsfälle des Systems beschrieben (entspricht den Detaillierungsgraden 1 und 2). Im dritten Kapitel „Specific Requirements“ finden sich die funktionalen und nicht-funktionalen Anforderungen, die dem Detaillierungsgrad 3 entsprechen.

1. Introduction
   - Purpose
   - Scope
   - Definitions
   - References
   - Overview
2. General description
   - Product perspective
   - Product functions
   - User characteristics
   - General constraints
   - Assumptions and Dependencies
3. Specific Requirements
   - Functional Requirements
   - External Interface Requirements
   - Performance Requirements
   - (...)
   - Other Requirements
4. Appendixes
5. Index

Abbildung 12.1: Referenzgliederung nach IEEE 830-1998 [4]

12.2.3 Qualitätskriterien für Anforderungen und das Anforderungdokument

Ziel des Requirements-Engineering ist es, ein Anforderungsdokument zu erstellen, dessen enthaltene Anforderungen genau das gewünschte System beschreiben. Damit dies erreicht werden kann, müssen die Anforderungen und das Anforderungsdokument bestimmten Qualitätskriterien genügen. In der Literatur werden viele verschiedene Kriterien genannt, von denen hier aus [5] ausgewählte vorgestellt werden.

Qualitätskriterien für die Anforderungen

- Vollständigkeit: Die gewünschte Funktionalität muss in ihrem gesamten Umfang beschrieben werden.
- Korrektheit: Die gewünschte Funktionalität muss genau so beschrieben werden, wie die Stakeholder sie sich vorstellen.
- Konsistenz: Eine Anforderung muss in sich und zu den anderen Anforderungen konsistent sein; es darf keine Widersprüche geben.
- Überprüfbarkeit: Es muss geprüft werden können, ob die Anforderung vom System umgesetzt wird.
- Verfolgbarkeit: Es muss nachvollziehbar sein, welche Anforderungen oder anderen Teile des Anforderungsdokuments von einer bestimmten Anforderung abhängig sind (auch unter dem Begriff *Traceability* bekannt).

Qualitätskriterien für das Anforderungsdokument

Das Anforderungsdokument muss aus Anforderungen bestehen, die den obigen Qualitätskriterien genügen. Zusätzlich muss das Anforderungsdokument selbst bestimmte Kriterien erfüllen:

- Sortierbarkeit: Das Anforderungsdokument muss so aufgebaut sein, dass es möglich ist, gezielt nur bestimmte Anforderungen herauszugreifen und zu lesen.
- Erweiterbarkeit: Das Anforderungsdokument muss so aufgebaut sein, dass während des Softwareentwicklungsprozesses Anforderungsänderungen ohne aufwändige Umstrukturierungen des Dokuments aufgenommen werden können.
- Angepasstheit: Das Anforderungsdokument muss so aufgebaut sein, dass es die Bedürfnisse des gegebenen Softwareentwicklungsprojekt optimal erfüllt.

12.2.4 Rollen im Requirements-Engineering

Für eine erfolgreiche Umsetzung des Requirements-Engineering in Softwareentwicklungsprojekten hat es sich nach [9] herausgestellt, dass für die Koordination des Requirements-Engineering-Prozesses die Einführung einer eigenen Rolle sinnvoll ist: die des *Requirements-Engineers*. Er gibt methodische Unterstützung bei der Erhebung der Anforderungen und übernimmt die Verwaltung der Anforderungen. Er vermittelt zwischen Anwendern und Entwicklern und ist dafür verantwortlich, dass alle relevanten Stakeholder in den Requirements-Engineering-Prozess eingebunden werden [2].

Der Requirements-Engineer unterscheidet sich vom *Projektleiter*, der die Verantwortung für das gesamte Projekt innehat [9]. Der Projektleiter dient bei Unstimmigkeiten und Schwierigkeiten als Vermittler zwischen dem Requirements-Engineer und dem Kunden.

Eine weitere im obigen Teil der Arbeit schon häufiger erwähnte Rolle ist der *Stakeholder*. Als Stakeholder werden alle Personen oder Institutionen verstanden, die in irgendeiner Weise von der Entwicklung oder vom Einsatz des Systems betroffen sind. Dazu gehören neben den offensichtlichen Stakeholdern, wie beispielsweise den Anwendern, die das System benutzen, den Entwicklern, die das System realisieren, und dem Kunden, der das System kauft, auch Institutionen, die die rechtliche Zulässigkeit des Systems prüfen, oder die Marketingabteilung, die für das System wirbt [2]. Den Stakeholdern kommt im Requirements-Engineering eine sehr wichtige Rolle zu, da hauptsächlich von ihnen die Anforderungen an das zu entwickelnde System ausgehen.

12.3 Aktivitäten im Requirements-Engineering

Der Prozess des Requirements-Engineering gliedert sich in die drei Aktivitäten *Anwendungsdomäne verstehen*, *Anforderungen erheben* und *Anforderungen managen*, die sich jeweils in einzelne Teilaktivitäten aufteilen [11]. Die in Abschnitt 12.1 genannten Aufgaben des Requirements-Engineering, das Erheben und Verwalten von Anforderungen, werden in den Aktivitäten *Anforderungen erheben* und *Anforderungen managen* umgesetzt. Bevor allerdings Anforderungen zielgerichtet erhoben werden können, müssen das Ziel und die Anwendungsdomäne des zu entwickelnden Systems bekannt sein [1]. Die Festlegung der Ziele und der Anwendungsdomäne findet in der Aktivität *Anwendungsdomäne verstehen* statt.

Im Folgenden werden die drei Aktivitäten im Detail vorgestellt, wobei auch auf eine Auswahl von Techniken eingegangen wird, mit Hilfe derer die Aktivitäten in der Praxis umgesetzt werden können. Gleichzeitig werden Empfehlungen gegeben, welche dieser Techniken für die PG geeignet sind. Die Beschreibung orientiert sich, wenn nicht anders angegeben, an [5].

12.3.1 Anwendungsdomäne verstehen

Ziel dieser Aktivität ist es, die Anwendungsdomäne des zu entwickelnden Systems zu beschreiben. Die Beschreibung beinhaltet alle Angaben, die den späteren Einsatzbereich des Systems spezifizieren. Dazu gehören sowohl Angaben zum *Systemkontext* und zu den relevanten *Stakeholdergruppen* als auch eine klare Zieldefinition des Projektes. Dieser Aktivität kommt im Requirements-Engineering eine große Bedeutung zu, da auf ihrer Grundlage die Anforderungen erhoben werden. Sind die Ziele der Systementwicklung ungenau formuliert oder sind wesentliche Stakeholdergruppen nicht erfasst, können das gesetzte Ziel nicht erreicht und die Anforderungen der fehlenden Stakeholder nicht umgesetzt werden. Dadurch wird die spätere Abnahme des Systems gefährdet.

Das Ergebnis der Aktivität *Anwendungsdomäne verstehen* sind grobe Anforderungen der Detaillierungsgrade 0 und 1 (siehe Abschnitt 12.2.1). In einem nach IEEE 830-1998 gegliederten Dokument würden diese Anforderungen in die ersten beiden Kapitel einfließen (siehe Abschnitt 12.1).

Die Aktivität *Anwendungsdomäne verstehen* kann in vier Teilaktivitäten untergliedert werden, die im Folgenden näher beschrieben werden. Diese können in beliebiger Reihenfolge durchgeführt werden, wobei es sich jedoch anbietet, zuerst die im Projekt angestrebten Ziele in einer *Vision* zu festzuhalten.

Vision erstellen

Die Vision enthält die Ziele, die mit dem zu entwickelnden System erreicht werden sollen [4]. Sie beschreibt das zu lösende Problem und geht darauf ein, *was*

das System zur Problemlösung beiträgt. Wichtig ist, dass die Vision lösungsneutral formuliert ist und keine Angaben darüber enthält, *wie* die Probleme gelöst werden sollen. Ansonsten würde der mögliche Lösungsraum zu früh eingeschränkt werden mit der Folge, dass gute Alternativlösungen nicht berücksichtigt würden.

Die Vision ist in einem Softwareentwicklungsprojekt sehr wichtig, da sie einen „Leitgedanken“ [4, S. 38] darstellt, der während der gesamten Entwicklung allen Stakeholdern zur Orientierung und gemeinsamen Ausrichtung verhilft.

Als Techniken, die bei der Erstellung einer Vision hilfreich sind, bieten sich nach [5] vor allem *Kreativitätstechniken* an, um sehr viele verschiedene Ideen zu sammeln. Beispielsweise können mit *Brainstorming* Ideen einer Gruppe von Personen unstrukturiert und unkommentiert gesammelt werden. Auch *Befragungen* geeigneter Personen aus den Stakeholdergruppen können genutzt werden, um die groben Zielvorstellung der Stakeholder zu sammeln.

In der PG sind alle diese Techniken einsetzbar, da sie allen Mitgliedern bekannt sind und sich ihr Durchführungsaufwand in einem vertretbarem Rahmen bewegt.

Systemkontext beschreiben

Als Systemkontext wird nach [5] der Teil der *Systemumgebung* bezeichnet, der Einfluss auf die Anforderungen an das System hat. Dazu gehören nicht nur mit dem System interagierende Systeme, sondern auch die Stakeholder und die Prozesse, in welche das System eingebettet ist. Wie in Abbildung 12.2 zu sehen, trennt die *Systemgrenze* den Systemkontext vom eigentlichen System.

Für die Anforderungserhebung ist es wichtig, dass die genaue Systemgrenze festgelegt ist, denn der Systemkontext beeinflusst das System und somit auch die Anforderungen an das System. An den Systemkontext selbst können allerdings keine Anforderungen gestellt werden, da er außerhalb des Systems liegt.

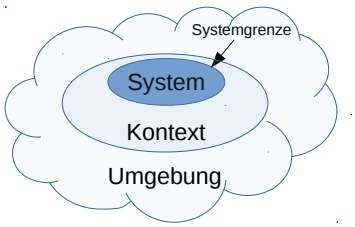


Abbildung 12.2: System, Systemumgebung und -kontext (nach [5])

Als Technik zum Festlegen des Systemkontextes bietet sich die Spezifikation von *Anwendungsfällen* an, um davon ausgehend Interaktionspartner des Systems und Schnittstellen zu ermitteln. Eine weitere Technik ist die Analyse der *Dokumentation*

externer Systeme, um genauere Anforderungen an die vom System zu verwendenden Schnittstellen zu erhalten.

Da das von der PG zu entwickelnde System Schnittstellen zu verschiedenen anderen Systemen haben wird, wird die Analyse der Dokumentation externer Systeme eine wichtige Rolle spielen. Die Spezifikation von Anwendungsfällen ist allen PG-Teilnehmern geläufig und kann somit ebenfalls verwendet werden.

Domänenmodell erstellen

Das Domänenmodell enthält nach [11] eine Übersicht über alle relevanten Stakeholdergruppen und spezifiziert den Anwendungsbereich des Systems in Form von Anwendungsfällen. Der vollständigen Erfassung aller Stakeholdergruppen kommt eine sehr große Bedeutung zu [5], da die Stakeholder die wichtigste Anforderungsquelle vor Dokumenten und existierenden (Konkurrenz-)Systemen darstellen. Ideen und Anforderungen nicht berücksichtigter Stakeholder werden bei der Systementwicklung nicht umgesetzt und werden sich nach der Auslieferung des Systems mit großer Wahrscheinlichkeit nachteilig auf den Absatz des Systems auswirken.

Da es nicht möglich ist, jeden einzelnen Stakeholder zu seinen Anforderungen an das System zu befragen, sollte aus jeder Stakeholdergruppe ein Repräsentant ausgewählt und im Domänenmodell dokumentiert werden, der in hohem Ausmaß für Rücksprachen mit den Entwicklern zur Verfügung steht [5]. Bei der Auswahl der Repräsentanten muss darauf geachtet werden, dass diese Personen in ihrer Stakeholdergruppe anerkannt sind, ausreichendes fachliches Wissen besitzen und dieses auch kommunizieren können.

Während in der Teilaktivität *Systemkontext beschreiben* die Anwendungsfälle verwendet wurden, um den Systemkontext und die Systemgrenzen zu bestimmen, sind beim Erstellen des Domänenmodells die Anwendungsfälle selbst relevant. Es bieten sich Befragungen von Stakeholdern als Technik an, um wesentliche Anwendungsfälle und weitere Stakeholder zu ermitteln. Diese Techniken sind, wie oben bereits erwähnt, in der PG anwendbar.

12.3.2 Anforderungen erheben

In der Aktivität *Anwendungsdomäne verstehen* wurden mit der Zieldefinition des Projekts und der Beschreibung des Anwendungsbereichs des Systems die Voraussetzungen dafür geschaffen, dass in der Aktivität *Anforderungen erheben* die Anforderungen an das System zielgerichtet erhoben werden können. Dabei werden zunächst viele verschiedene Wünsche an das System *ermittelt* und anschließend in geeigneter Weise als Anforderungen *dokumentiert*. Es folgt eine *Validierung*, ob die Anforderungen den in Abschnitt 12.2.3 genannten Qualitätskriterien genügen. Zum Schluss müssen die Stakeholder bei konfliktären Anforderungen *verhandeln*, um eine Lösung zu erzielen, die von allen Stakeholdern *verabschiedet* wird.

Anforderungen ermitteln

In dieser Teilaktivität werden alle bekannten Wünsche an das System gesammelt. Die nächstliegendste Anforderungsquelle sind die Stakeholder, vor allem die späteren Anwender. Darüber hinaus können nach [2] auch ähnliche, bereits existierende Konkurrenzsysteme oder, im Fall eines Softwareevolutionsprojektes, das zu ersetzende Altsystem als Anforderungsquelle dienen. Die Analyse solcher Systeme und deren Dokumentation kann Auskunft über mögliche Schwachstellen und Erfolgsfaktoren der Systeme geben. Falls das zu entwickelnde System einen fremden Fachbereich berührt, ist das Studium von Fachliteratur eine nützliche Anforderungsquelle, um beispielsweise in der Domäne verwendete Algorithmen kennenzulernen. Soll das System bestimmte Standards erfüllen, sind auch diese eine zu berücksichtigende Anforderungsquelle.

Als Techniken für die Ermittlung von Anforderungen können außer den vier in Abschnitt 12.3.1 genannten Techniken Befragung, Anwendungsfälle, Kreativitätstechniken und Analyse bestehender Systeme unter anderem auch die *Szenario-Methode* und *Anforderungs-Workshops* eingesetzt werden. Die Szenario-Methode hat zum Ziel, die grobe Funktion des Systems mit Hilfe einer Menge von Szenarien zu beschreiben [5]. Ein Szenario beschreibt den Ablauf einer konkreten Interaktion eines Benutzers mit dem System. Diese Methode eignet sich insbesondere dann, wenn Anforderungen von Stakeholdern erhoben werden sollen, die kein ausgeprägtes Abstraktionsvermögen besitzen.

In einem Anforderungsworkshop werden gemeinsam mit allen Repräsentanten relevanter Stakeholdergruppen Anforderungen ausgehend von einer groben Ebene schrittweise verfeinert. Während des Workshops können wiederum verschiedene Techniken angewendet werden, auf die an dieser Stelle nicht weiter eingegangen werden soll. Für eine detailliertere Beschreibung siehe [6]. Der Vorteil eines solchen Anforderungsworkshops ist, dass die Stakeholder ihre unterschiedlichen Sichtweisen auf das System und die einzelnen Anforderungen frühzeitig gemeinsam in Übereinstimmung bringen können.

Für die PG ist die Szenario-Methode wahrscheinlich weniger relevant, da bei dem zu entwickelnden System die Benutzerinteraktion eine untergeordnete Rollen spielen wird. Ein Anforderungsworkshop hingegen ist zu empfehlen, da auf diese Weise alle PG-Beteiligten gemeinsam in die Anforderungserhebung einbezogen werden.

Anforderungen dokumentieren

Während das Ergebnis der Teilaktivität *Anforderungen ermitteln* eine Sammlung unstrukturierter Wünsche an das System ist, liefert die Teilaktivität *Anforderungen dokumentieren* ein strukturiertes vorläufiges Anforderungsdokument, das einheitlich formulierte Anforderungen enthält. Für die Dokumentation von Anforderungen gibt es verschiedene Techniken: *grafisch*, *natürlichsprachlich*, *semi-formal* und *formal*.

Die grafische Notation in Form von UML-Diagrammen eignet sich nach [5] besonders dafür, Anforderungen übersichtlich und präzise darzustellen. Allerdings wer-

den UML-Diagramme bei komplexen Zusammenhängen oft unübersichtlich. In diesem Fall sollte ein übersichtlich gehaltenes UML-Diagramm mit ins Detail gehenden, natürlichsprachlichen Anforderungen kombiniert werden, obwohl natürlichsprachliche Anforderungen von allen vier Techniken den größten Interpretationsspielraum bieten. Um diesen zu verringern, schlägt [5] eine Schablone vor, nach der alle natürlichsprachlichen Anforderungen formuliert werden (siehe Abbildung 12.3). Eine nach der Schablone formulierte Anforderung ist beispielsweise: „Nachdem der Benutzer sein Passwort eingegeben hat, muss das System dem Benutzer die Möglichkeit bieten, den Kontostand abzurufen."

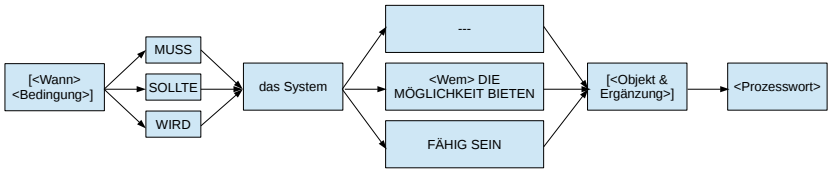


Abbildung 12.3: Schablone für natürlichsprachliche Anforderungen [5, S. 166]

Eine semi-formale Dokumentationstechnik sind *Use-Case-Beschreibungen*, die sich eignen, um Use-Case-Diagramme zu verfeinern. Damit können grobe Anforderungen natürlichsprachlich, aber in einem formalisierten Rahmen beschrieben werden. [5] schlägt vor, Use-Cases-Beschreibungen tabellarisch zu notieren, wie in Tabelle 12.1 gezeigt ist.

Die formale Dokumentationstechnik mit Hilfe von mathematischen Ausdrücken erlaubt eine sehr präzise Formulierung der Anforderungen, ist aber für die meisten Stakeholder schwer zu verstehen. Sie wird nach [3] von den vorgestellten Techniken in der Praxis am wenigsten verwendet.

Bei der Auswahl und Kombination der Dokumentationstechniken sollte insbesondere auf die Akzeptanz der jeweiligen Technik bei den Stakeholdern geachtet werden. Zusätzlich sollte berücksichtigt werden, dass, je mehr verschiedene Techniken verwendet werden, der Aufwand für die Sicherstellung der Konsistenz und Verfolgbarkeit der Anforderungen steigt.

Die dargestellten Dokumentationstechniken werden von allen PG-Beteiligten verstanden. Da allerdings eine formale Dokumentation der Anforderungen nicht notwendig sein wird, sollte in der PG eine Kombination aus übersichtlichen UML-Diagrammen, die bei Bedarf mit Use-Case-Beschreibungen verfeinert werden können, und detaillierten natürlichsprachlichen Anforderungen gewählt werden.

Anforderungen validieren

Das in der vorigen Teilaktivität erstellte vorläufige Anforderungsdokument muss validiert werden, bevor mit der Entwicklung des Systems begonnen werden kann.

Name	Leihobjekt verleihen
Kurzbeschreibung	Der Use-Case beschreibt, welche Prüfungen in welcher Reihenfolge nötig sind, dass ein Kunde ein Leihobjekt aus der Bibliothek ausleihen kann.
Akteure	Kunde, Bibliothekar
Vorbedingungen	Der Bibliothekar ist am System angemeldet.
Fachlicher Auslöser	Der Kunde übergibt das Leihobjekt dem Bibliothekar.
Normalfall	Kunde identifizieren Entleihberechtigung prüfen (...) Ausleihe speichern

Tab. 12.1: Schablone für Use-Case Beschreibungen [5, S. 215]

Dazu muss geprüft werden, ob die einzelnen Anforderungen und das gesamte Anforderungsdokument alle Qualitätskriterien erfüllen (siehe Abschnitt 12.2.3).

Eine weit verbreitete Technik sind *Reviews*, die auf dem Prinzip beruhen, dass neben dem Verfasser auch andere Personen die Anforderungen lesen und ihre Qualität prüfen. Reviews können nach [5] anhand ihres Formalisierungsgrades unterschieden werden in

- *Stellungnahme*: wenig formal; der Verfasser gibt einem Kollegen die Anforderungen zum Lesen
- *Walkthrough*: etwas formal; der Verfasser stellt in einer Sitzung mit allen Projektbeteiligten die von ihm formulierten Anforderungen vor
- *Inspektion*: sehr formales Verfahren mit mehreren definierten Phasen

Eine weitere Technik zur Validierung von Anforderungen sind *Testfälle*. Sie haben den positiven Effekt, dass der Entwickler der Testfälle sich sehr ausführlich mit allen Anforderungen auseinandersetzen muss und auf diese Weise viele Fehler im Anforderungsdokument entdecken kann. Ist beispielsweise für eine Anforderung kein Testfall erstellbar, ist dies ein starkes Indiz dafür, dass die Anforderung das Qualitätskriterium der Überprüfbarkeit nicht erfüllt.

Als weitere Validierungstechnik können *Prototypen* eingesetzt werden, die einen Teil der gewünschten Funktionalität umsetzen. Prototypen ermöglichen den Stakeholdern, das System bereits zu einem frühen Zeitpunkt in der Praxis zu erleben und ihre Anforderungen anschließend konkreter zu formulieren. Diese Methode ist nach [4] die effektivste, um Fehler in den Anforderungen zu finden.

In der PG bieten sich aufgrund des geringeren Vorbereitungs- und Durchführungsaufwand die weniger formalen Review-Techniken an. Diese könnten kombiniert werden, indem durch Stellungnahmen einzelne Teile der Anforderungen validiert werden, bevor in einem Walkthrough mit allen Beteiligten die Gesamtheit der Anforderungen validiert wird. Die frühzeitige Entwicklung von Testfällen und Prototypen sollte auf Grund ihrer oben genannten Vorteile ebenfalls angestrebt werden.

Anforderungen verhandeln

Stellt sich bei der Validierung der Anforderungen heraus, dass Anforderungen konfliktär sind, müssen die Stakeholder diese Widersprüche auflösen. Für die Konfliktlösung schlägt [4] verschiedene Techniken vor, von denen hier drei herausgegriffen werden. Die *Entscheidung* ist eine Technik, bei der eine Person mit entsprechender Kompetenz festlegt, welche Lösung die endgültige ist. Die *Abstimmung* ist eine demokratische Entscheidung, bei der die Mehrheit den Ausschlag gibt. Die sinnvollste Methode ist die *Einigung*, bei der die Stakeholder durch Argumentation ihre Standpunkte darlegen und insofern zu einem Konsens führen, als dass eine der beiden Meinungen akzeptiert wird oder ein Kompromiss erzielt wird. Diese Technik sollte nach Möglichkeit auch in der PG zum Einsatz kommen.

Anforderungen verabschieden

Sind alle während der Anforderungsvalidierung aufgetretenen Konflikte gelöst, müssen alle Stakeholder das Anforderungdokument verabschieden, um auf diese Weise ihr Einverständnis mit den dokumentierten Anforderungen zu erklären. Mit der Verabschiedung des Anforderungsdokuments ist die Grundlage für die nachfolgende Entwicklung des Systems gegeben.

12.3.3 Anforderungen managen

Auch wenn zu Beginn eines Softwareentwicklungsprojektes das Anforderungsdokument von allen Stakeholdern verabschiedet wurde, ergeben sich während fast jedes Projektes Anforderungsänderungen. Die Ursachen hierfür können nach [6] unter anderem Fehler im Anforderungsdokument oder Änderungen im Systemkontext sein. Eine weitere Ursache ist, dass die Kunden ihre Anforderungen während des Projektverlaufs konkretisieren oder ändern, weil ihnen beispielsweise erst ein Prototyp eine klare Vorstellung von dem späterem System ermöglichte.

Die Aktivität *Anforderungen managen*, auch als *Requirements-Management* bezeichnet, dient der Verwaltung der Anforderungen und Anforderungsänderungen während der gesamten Projektlaufzeit und stellt somit eine Querschnittsdisziplin des Requirements-Engineering dar. Nach [5] gehört es zur Aufgabe des Requirements-Management, ein *Requirements-Engineering-Konzept* zu entwickeln, in dem „alle Prozesse, Methoden, Tools, Rollen und Vorgehensweisen dokumentiert sind, die für die Durchführung der Anforderungsanalyse gebraucht werden“ [5, S. 346]. Das Requirements-Engineering-Konzept sollte so ausgestaltet sein, dass die Auswirkungen von Anforderungsänderungen so gering wie möglich sind [6]. Zur Aufgabe des Requirements-Management gehört es ebenfalls, die Umsetzung des Requirements-Engineering-Konzeptes zu kontrollieren [5].

[5] macht verschiedene Vorschläge, wie die Verwaltung der Anforderungen unterstützt werden kann. So hat es sich als hilfreich erwiesen, die Struktur des Dokuments erweiterbar anzulegen, sodass Kapitel hinzugefügt oder verschoben werden können, ohne dass das gesamte Dokument umstrukturiert werden muss. Die in Abschnitt 12.2.3 als Qualitätskriterium geforderte Traceability zwischen Anforderungen kann hergestellt werden, indem die Anforderungen Identifikatoren erhalten, die während des gesamten Projektes eindeutig sind und sich nicht ändern. Außerdem macht [5] den Vorschlag, Anforderungen um Status-Attribute zu ergänzen, die kennzeichnen, ob eine Anforderung bereits umgesetzt, in Bearbeitung oder noch offen ist. Damit wird die Messung des Projektfortschritts ermöglicht. Diese Vorschläge sollten auch in der PG umgesetzt werden.

12.4 Einbettung des Requirements-Engineering in Vorgehensmodelle

Die bisherige Betrachtung des Requirements-Engineering erfolgte unabhängig von einem konkreten Vorgehensmodell. In diesem Abschnitt soll nun kurz darauf eingegangen werden, wie das Requirements-Engineering in die drei Vorgehensmodelle *Wasserfallmodell*, *Scrum* und *Unified Process (UP)* eingebettet ist. Aufgrund des beschränkten Umfangs der Arbeit wird an dieser Stelle auf eine Einführung dieser Vorgehensmodelle verzichtet.

Im Wasserfallmodell findet zu Beginn eine einmalige Analysephase statt, während der alle Anforderungen an das zu entwickelnde System erhoben werden. Allerdings ist im Wasserfallmodell kein Requirements-Management vorgesehen, sodass Anforderungsänderungen während der folgenden Phasen zu Problemen führen. Diesem Mangel begegnen agile Vorgehensmodelle mit einem Entwicklungsprozess, der auf die kontinuierliche Anforderungserhebung und -verwaltung ausgelegt ist [6].

In dem agilen Vorgehensmodell Scrum werden die Anforderungen „just-in-time“ erhoben [2, S. 321], das heißt genau dann, wenn sie benötigt werden. Das Requirements-Engineering findet während der Sprintplanung (Auswahl der im Sprint zu realisierenden User Stories) und während des Sprint Reviews (Anpassung des Product Backlogs an die Wünsche des Product Owners) statt. Das Requirements-Management ist in Scrum indirekt dadurch integriert, dass Anforderungsänderungen während eines Sprints in das Product Backlog aufgenommen werden.

Unified Process ist ein iteratives Vorgehensmodell, das definierte Meilensteine für die Dokumentation des Projektziels und die Anforderungen festlegt. Das Requirements-Engineering und Requirements-Management sind explizit durch einen eigenen Workflow integriert und finden zwar zu Beginn des Projektes verstärkt, aber kontinuierlich während des gesamten Projektes statt.

12.5 Empfehlungen für die PG

Nach Betrachtung des generellen Ablaufs des Requirements-Engineering und möglicher einzusetzender Techniken stellt sich die Frage, wie in der PG, in deren Kontext die vorliegende Arbeit entstanden ist, ein zweckerfüllendes Requirements-Engineering ausgestaltet werden kann. In diesem Abschnitt werden Empfehlungen zu den nächsten Schritten der PG bezüglich des Requirements-Engineering gegeben.

In der PG sollte sobald wie möglich die Rolle des Requirements-Engineer besetzt werden, damit dieser das Requirements-Engineering-Konzept entwerfen und die weiteren Schritte des Requirements-Engineering koordinieren kann. Außerdem sollte sich die PG in nächster Zeit für ein Vorgehensmodell entscheiden, um das Requirements-Engineering-Konzept an dieses anpassen zu können. Am besten geeignet wäre vermutlich eine Mischung aus einem agilen und einem phasenorientierten Vorgehensmodell. Denn während für die Entwicklungsphase des Projektes agile Verfahren gut geeignet sind, um schnell Ergebnisse zu produzieren und auf Anforderungsänderungen direkt einzugehen, ist zu Beginn des Projektes eine Analysephase notwendig, in der die wichtigsten Anforderungen an das System erhoben werden, um eine zielgerichtete Entwicklung zu ermöglichen.

Die PG sollte unabhängig vom gewählten Vorgehensmodell mit einer Analysephase beginnen, um zunächst die Anwendungsdomäne zu verstehen (Aktivität *Anwendungsdomäne verstehen*, Abschnitt 12.3.1) und grobe Anforderungen an das zu entwickelnden Systems zu erheben (Aktivität *Anforderungen erheben*, Abschnitt 12.3.2). Für die PG bietet es sich an, diese Aktivitäten nicht nur einmalig zu Beginn des Projektes durchzuführen, sondern zunächst grobe Anforderungen zu erheben und diese in weiteren Iterationen zu verfeinern. Ein solches Vorgehen ist sinnvoll, weil den PG-Mitgliedern nicht alle Anforderungen und Anforderungsquellen zu Beginn des Projektes bekannt sind und sie sich im Laufe des Projekts Wissen in der zunächst noch fremden Domäne aneignen werden, welches dann ebenfalls in die Anforderungen einfließen wird. Ein solches Vorgehen wird auch von [4] vorgeschlagen. Die PG könnte das Vorgehen in der Praxis umsetzen, indem beispielsweise in einer ersten Iteration durch das Studium von Fachliteratur und Interviews ein Überblick über die relevanten Funktionen des Systems geschaffen wird und in einer zweiten Iteration während eines Anforderungsworkshops detailliertere Anforderungen erhoben werden.

12.6 Zusammenfassung

In der vorliegenden Arbeit wurde deutlich, dass das Requirements-Engineering in einem Softwareentwicklungsprojekt für die zielgerichtete Systementwicklung und ein Requirements-Engineer für die Koordination des Requirements-Engineering notwendig sind. Es wurde beschrieben, dass sich das Requirements-Engineering in die drei Aktivitäten *Anwendungsdomäne verstehen*, *Anforderungen erheben* und *Anforderungen managen* gliedert, die in der Praxis mit Hilfe verschiedener Techniken

unterstützt werden können. Außerdem wurde erwähnt, dass die Wahl des Vorgehensmodells die Art und Weise der Integration des Requirements-Engineering in das Softwareentwicklungsprojekt stark beeinflusst.

Für die PG wurde die Empfehlung gegeben, mit einer Analysephase zu beginnen, um die Anwendungsdomäne zu verstehen und grobe Anforderungen zu erheben. Diese Anforderungen sollten stabil genug sein, um den Beginn der Implementierungsphase zu ermöglichen. Dabei sollten die Anforderungen iterativ verfeinert werden, um den Wissenszuwachs der PG-Beteiligten zu berücksichtigen. Von den im ersten Teil der Arbeit vorgestellten Techniken sollten diejenigen angewendet werden, die den PG-Beteiligten bekannt und mit vertretbarem Aufwand umsetzbar sind.

Literaturverzeichnis

[1] Dines Bjørner. *Software Engineering 3*. Berlin Heidelberg: Springer, 2006.

[2] Ulrike Hammerschall und Gerd Beneken. *Software Requirements*. München: Pearson, 2013.

[3] Colin J. Neill und Phillip A. Laplante. „Requirements Engineering: The State of the Practice“. In: *IEEE Software*. Bd. 20, no. 6. IEEE, November/December 2003. URL: %7Bhttp://www.personal.psu.edu/cjn6/Personal/Reprints/Software%2020v20n6%202003.pdf%20(20.05.15)%7D.

[4] Klaus Pohl. *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Heidelberg: dpunkt, 2007.

[5] Chris Rupp. *Requirements-Engineering und -Management*. 5. Aufl. München: Carl Hanser, 2009.

[6] Bruno Schienmann. *Kontinulierliches Anforderungsmanagement. Prozesse - Techniken - Werkzeuge*. München: Addison-Wesley, 2002.

[7] Ian Sommerville. *Software-Engineering*. 9. Aufl. Boston: Pearson, 2011.

[8] Standish Group. *CHAOS MANIFESTO 2013. Think Big, Act Small*. 2013. URL: %7Bhttp://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf%20(14.05.15)%7D.

[9] Gerhard Versteegen, Hrsg. *Anforderungsmanagement*. Berlin Heidelberg: Springer, 2004.

[10] Karl E. Wiegers. *Software Requirements. Practical techniques for gathering and managing requirements throughout the product development cycle*. Redmond: Microsoft Press, 2003.

[11] Andreas Winter. *Vorlesungsfolien Requirements-Engineering und -Management*. Wintersemester 2014/15.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungen

BDEW	Bundesverband der Energie- und Wasserwirtschaft e.V.
BHKW	Blockheizkraftwerk
BUC	Business Use Case (BUC)
CI	kontinuierliche Integration
CIM	Common Information Model
CMMI	Capability Maturity Model Integration
EEG	Erneuerbare-Energien-Gesetz
EPEX SPOT	European Power Exchange Spotmarket
GUI	grafische Benutzeroberfläche
HLUC	High Level Use Case
k-NN	K-nearest neighbors
OPC UA	OPC Unified Architecture
PUC	Primary Use Case
PV-Anlage	Photovoltaik-Anlage
RMI	Remote Method Invocation
RPC	Remote Procedure Call
ScrUP	Scrum Unified Process
SG-Plane	Smart Grid Plane
SGAM	Smart Grid Architecture Model
WXXM	Weather Information Exchange Model

Literatur

[AM12] ANDERSEN, ANDRE CHRISTOFFER und STIAN MIKELSEN: *A Novel Algorithmic Trading Framework Applying Evolution and Machine Learning for Portfolio Optimization*. Masterthesis, Norwegische Universität für Wissenschaft und Technologie, 2012.

[Bag02] BAGEMIHL, JOACHIM: *Optimierung eines Portfolios mit hydro-thermischem Kraftwerkspark im börslichen Strom- und Gasterminmarkt*. Doktorthesis, Universität Stuttgart, 2002.

[BDE07] BDEW BUNDESVERBAND DER ENERGIE- UND WASSERWIRTSCHAFT E.V.: *Praxisinformation P 2007 / 13 Gastransport / Betriebswirtschaft Abwicklung von Standardlastprofilen*, 2007.

[BDE13] BDEW BUNDESVERBAND DER ENERGIE- UND WASSERWIRTSCHAFT E.V.: *Erneuerbare Energien und das EEG: Zahlen, Fakten, Grafiken (2013)*, 2013.

[BDE15] BDEW BUNDESVERBAND DER ENERGIE- UND WASSERWIRTSCHAFT E.V.: *BDEW/VKU/GEODE Leitfaden Abwicklung von Standardlastprofilen Gas*, 2015.

[BLB+16] BUITINCK, LARS, GILLES LOUPPE, MATHIEU BLONDEL, FABIAN PEDREGOSA, ANDREAS MUELLER, OLIVIER GRISEL, VLAD NICULAE, PETER PRETTENHOFER, ALEXANDRE GRAMFORT, JAQUES GROBLER, ROBERT LAYTON, JAKE VANDERPLAS, ARNAUD JOLY, BRIAN HOLT und GAËL VAROQUAUX: *RBF SVM parameters*. http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html, März 2016.

[BMW] *Gesetz für den Ausbau Erneubarer Energien. Erneuerbare-Energien-Gesetz-2014*.

[Bre15] BREMER, JÖRG: *Constraint-Handling mit Supportvektor-Dekodern in der verteilten Optimierung*. Doktorarbeit, 2015.

[BS] BERNHARD, RALPH und ACHIM SZTUKA: *Projektmanagement - Überblick über Projektphasen und Projektorganisation*. http://www.manager-wiki.com/methodik/59-projektmanagement. Zuletzt besucht am 01.03.2016.

[BSfWuM] MEDIEN, ENERGIE UND TECHNOLOGIE BAYERISCHES STAATSMINISTERIUM FÜR WIRTSCHAFT UND: *Energie-Atlas Bayern Daten und Fakten*. https://www.energieatlas.bayern.de/thema_sonne/photovoltaik/daten.html. Zuletzt besucht am 31.03.2016.

[BSW] *EEG 2014 – Erlösobergrenzen im Sinne des Marktprämienmodell*.

[Buc] BUCHHOLZ, PETER: *Modellgestützte Analyse und Optimierung. Vorlesungsskript Technische Universität Dortmund*. http://ls4-www.cs.tu-dortmund.de/cms/

de/lehre/vorherige_semester/2013_ss/mao/index.html. Zuletzt besucht am 30.11.2015.

[Bun] BUND DER ENERGIE VERBRAUCHER: *Aktuelle regionale Fluessiggas-Preise*. http://www.energieverbraucher.de/de/preisabfrage__1101/. Zuletzt besucht am 16.03.2016.

[CD00] CHENG, JIAN und MAREK J. DRUZDZEL: *Latin Hypercube Sampling in Bayesian Networks*. www.pitt.edu/~druzdzel/psfiles/flairs00.pdf, 2000. Zuletzt besucht am 04.03.2016.

[Drü12] DRÜCK, DR.-ING. HARALD: *Solarthermie I Teil 1 Vorlesung Universität Stuttgart*, 2012.

[Eis05] EISENBACH, DOMINIK: *Künstliche Neuronale Netze zur Prognose von Zeitreihen*. Diplomarbeit, Westfälische Wilhelms-Universität Münster, 2005.

[EM03] ESSIGKRUG, ANDREAS und THOMAS MEY: *Rational Unified Process kompakt*. Spektrum Akademischer Verlag, 2003.

[Eur16] EUROPEAN POWER EXCHANGE (EPEX SPOT SE): *EPEX SPOT Operational Rules*, 2 2016. Accesable at EPEX SPOT Market Rules and Regulation : http://www.epexspot.com/en/extras/download-center.

[Fraa] FRANK, ALEXANDER: *2K-teilfaktorielle Versuchspläne (Vorlesungsfolien)*. www.tqu-group.com/vorlesungen/VorlFrank/HS_3-05%202k%20teilfaktorielle%20Versuche.pdf. Zuletzt besucht am 04.03.2016.

[Frab] FRANK, ALEXANDER: *2K-vollfaktorielle Versuchspläne (Vorlesungsfolien)*. www.tqu-group.com/vorlesungen/VorlFrank/HS_3-04%202k%20vollfaktorielle%20Versuche.pdf. Zuletzt besucht am 04.03.2016.

[FST06] FUCHS, A., T. SCHWEINFURTH und M. THIEL: *Energie-Agenda Allensbach*, 2006.

[fWuA] AUSFUHRKONTROLLE, BUNDESAMT FÜR WIRTSCHAFT UND: *Stromvergütung für KWK-Anlagen*. http://www.bafa.de/bafa/de/energie/kraft_waerme_kopplung/stromverguetung/index.html. Zuletzt besucht am 30.03.2016.

[Gie15] GIEßSEN, J.: *Physik und Astronomie*. www.geoastro.de/SME/tk/index.html, 2015.

[GKLL+03] GÖWE-KUSKA, DR. N., DR. A. LIEBSCHER, DIPL.-ING. M. LUCHT, PROF. DR. W. RÖMISCH, DR. G. SPANGARDT und DIPL.-MATH. I. WEGNER: *Mittelfristige risikoorientierte Optimierung von Strombeschaffungs-Portfolios kleinerer Marktteilnehmer*. Konferenzpaper, Humboldt–Universität Berlin and DREWAG Dresden and Fraunhofer-Institut UMSICHT Oberhausen, 2003.

[Glo13] GLOGER, BORIS: *Scrum: Produkte zuverlässig und schnell entwickeln*. Carl Hanser Verlag, 4. Auflage, 2013.

[Gro13] GROUP, CEN-CENELEC-ETSI SMART GRID COORDINATION: *OCEN-CENELEC-ETSI Smart Grid Coordination Group Smart Grid Reference Architecture*. CEN-CENELEC-ETSI Smart Grid Coordination Group, ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf, November 2013.

[Hei] HEIZUNGSFINDER: *Anschaffungskosten und Betriebskosten eines BHKW*. www.heizungsfinder.de/bhkw/kosten-preise. Zuletzt besucht am 16.03.2016.

[Hes] HESSISCHES MINISTERIUM FÜR UMWELT, ENERGIE, LANDWIRTSCHAFT UND VERBRAUCHERSCHUTZ: *Niedertemperatur- und Brennwertkessel. Wissenswertes über moderne Zentralheizungsanlagen (2011)*. www.gemeinde-fuerth.de/gv_fuerth/Gesundheit,%20Soziales,%20Umwelt/Umwelt%20und%20Energie/Energieberatung/Niedertemperatur-_und_Brennwertkessel_04-11.pdf. Zuletzt besucht am 10.09.2015.

[hLHrnhMS15] L. HOFMANN, PROF. DR.-ING. HABIL. und PROF. DR. RER. NAT. HABIL. M. SONNENSCHEIN: *Smart Nord Final Report*. Forschungsprojekt, Universität Oldenburg und Universität Hannover, 2015.

[Jor12] JORDAN, DR. U.: *Vorlesung Solarthermie Vorlesung Universität Kassel*, 2012.

[Kel12] KELLER, FLORIAN: *Konzepte des maschinellen Lernens*. Masterthesis, Fachhochschule Köln, 2012.

[KPP04] KELLERER, HANS, ULRICH PFERSCHY und DAVID PISINGER: *Knapsack problems*. Springer, 2004.

[Lin10] LIN, CHIH-WEI HSU CHIH-CHUNG CHANG CHIH-JEN: *A Practical Guide to Support Vector Classification*. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, April 2010.

[Mit15] MITTELMANN, HANS: *DECISION TREE FOR OPTIMIZATION SOFTWARE*. http://plato.la.asu.edu/bench.html, 2015.

[Mol08] MOLTER, DR. K.: *Vorlesungsskript Alternative Energietechnik*. Technischer Bericht, FH Trier, 2008.

[Neu14a] NEUMANN, ALEXANDER: *SparxSystems CE: Intelligentes Stromnetz modellbasiert entwickeln*. http://www.heise.de/developer/meldung/Angewandte-Forschung-Intelligentes-Stromnetz-modellbasiert-entwickeln-2195432.html, 05 2014.

[Neu14b] NEUREITER, CHRISTIAN: *Introduction to the „SGAM Toolbox“*, April 2014.

[NEX15] NEXT KRAFTWERKE: *Marktprämie & Marktprämienmodell*, 2015.

[Nie15] NIEßSE, ASTRID: *Verteilte kontinuierliche Einsatzplanung in Dynamischen Virtuellen Kraftwerken*. Doktorarbeit, Universität Oldenburg, 2015.

[NISa] NIST/SEMATECH: *e-Handbook of Statistical Methods: Full factorial designs*. http://www.itl.nist.gov/div898/handbook/pri/section3/pri333.htm. Zuletzt besucht am 04.03.2016.

[NISb] NIST/SEMATECH: *e-Handbook of Statistical Methods: What is experimental design?* http://www.itl.nist.gov/div898/handbook/pri/section1/pri11.htm. Zuletzt besucht am 04.03.2016.

[NISc] NIST/SEMATECH: *e-Handbook of Statistical Methods: Yates Algorithm*. www.itl.nist.gov/div898/handbook/eda/section3/eda35i.htm. Zuletzt besucht am 16.03.2016.

[NT16] NETZ-TRANSPARENZ: *Marktwert- / Referenzmarktwertübersicht*, 2016.

[OPC16a] OPC FOUNDATION: *OPC Unified Architecture Specification - Part 1: Overview and Concepts*. OPC Foundation, http://www.opcfoundation.org/UA/Part1/, März 2016.

[OPC16b] OPC FOUNDATION: *Unified Architecture*. https://opcfoundation.org/about/opc-technologies/opc-ua/, März 2016.

[Ora] ORACLE: *JavaFX Frequently Asked Questions*. http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html#5. Zuletzt besucht am 24.03.2016.

[Pho15] PHOTOVOLTAIK.ORG: *Marktprämie*. http://www.photovoltaik.org/wissen/marktpraemie, 2015.

[Pro] PROJEKTMAGAZIN: *Glossar Vision*. https://www.projektmagazin.de/glossarterm/vision. Zuleezt besucht am 25.03.2016.

[PTK15] POLOCZEK, JENDRIK, NILS ANDRE TREIBER und OLIVER KRAMER: *KNN Regression as Geo-Imputation Method for Spatio-Temporal Wind Data*. Scientific Paper, Universität Oldenburg, 2015.

[Qua11] QUASCHING, V.: *Regenerative Energiesysteme: Technologie - Berechnung - Simulation*. Carl Hanser Verlag GmbH & Co. KG, 2011.

[SH13] STIEGLITZ, R. und V. HEINZEL: *Thermische Solarenergie: Grundlagen, Technologie, Anwendungen*. Springer, 2013.

[Sie16] SIEMENS AKTIENGESELLSCHAFT: *IEC 61850*. http://m.energy.siemens.com/hq/de/energy-themen/normen/iec61850.htm, März 2016.

[SM11] SOLIMAN, SOLIMAN ABDEL-HADY und ABDEL-AAL HASSAN MANTAWY: *Modern Optimization Techniques with Applications in Electric Power Systems* -. Springer Science and Business Media, Berlin Heidelberg, 2012. Aufl. Auflage, 2011.

[Spl02] SPLIID, HENRIK: *Design and Analysis af Experiments with k Factors having p Levels. Lecture notes in the Design and Analysis of Experiments. Technical University of Denmark, Lyngby*. http://www2.imm.dtu.dk/courses/02411/engnote.pdf, 2002. Zuletzt besucht am 16.03.2016.

[UD10] UNGER, THOMAS und STEPHAN DEMPE: *Lineare Optimierung: Modell, Lösung, Anwendung*. Vieweg+Teubner Verlag / GWV Fachverlage GmbH, 2010.

[Usl16] USLAR, DR.-ING. MATHIAS: *IEC 61850 - Feldebenenkommunikation*. Vorlesung, Januar 2016.

[USR+13] USLAR, MATHIAS, MICHAEL SPECHT, SEBASTIAN ROHJANS, JÖRN TREFKE, CHRISTIAN DÄNEKAS, JOSÉ M. GONZÁLEZ, CHRISTINE ROSINGER und ROBERT BLEIKER: *Standardization in Smart Grids - Introduction to IT-related Methodologies, Architectures and Standards*. Juli, 2013.

[vO12] OEHSEN, AMANY VON: *Entwicklung und Anwendung einer Kraftwerks- und Speichereinsatzoptimierung für die Untersuchung von Energieversorgungsszenarien mit hohem Anteil erneuerbarer Energien in Deutschland*. Doktorthesis, Universität Kassel, 2012.

[Was05] WASLANDER, STEVEN: *A Primer on Mixed Integer Linear Programming. Using Matlab, AMPL and CPLEX at Stanford University*. web.stanford.edu/class/aa278a/AA278A_MILP_Primer.ppt, Mai 2005. Zuletzt besucht am 29.03.2016.

[Wek] *Time Series Analysis and Forecasting with Weka*. http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka. Zuletzt besucht am 29.03.2016.

Glossar

Powder

Powder ist das Akronym für *Profit Optimization With Distributed Energy Resources*. Die Entstehung des Begriffs kann in Kapitel 2.2 nachgelesen werden.

Anlage

Als Anlage wird die Aggregation von Bauteilen eines (dezentralen) Energieerzeugers als ein System bezeichnet. Eine Anlage kann auch Komponenten von PV, BHKW usw. zusammenfassen.

Anlagenbetreiber

Der Anlagenbetreiber ist der Besitzer bzw. Betreiber einer Anlage.

Anlagenkonfiguration

Als Anlagenkonfiguration wird die Zusammenstellung der ausgewählten parametrisierten Anlagentypen bezeichnet. Sie enthält für jeden parametrisierten Anlagentyp dessen Name und die verwendete Anzahl an Anlagen (s. Abschnitt 6.4.1).

Anlagentyp

Die Anlagen in *Powder* sind vom Typ PV oder BHKW.

Bamboo

Bamboo ist ein Server für kontinuierliche Integration von der Firma Atlassian.

Blockheizkraftwerk

Blockheizkraftwerke sind Energieerzeugungsanlagen, die das Prinzip Kraft-Wärme-Kopplung umsetzen. Sie besitzen einen modularen Aufbau. Blockheizkraftwerke verwenden für die Strom- und Wärmeerzeugung Verbennungsmotoren, Dieselmotoren oder Gasturbinen und gehören zu den flexiblen dezentralen Erzeugungsanlagen.

Bundesverband der Energie- und Wasserwirtschaft e.V.

Der Bundesverband der Energie- und Wasserwirtschaft e.V. ist ein einflussreicher Interessenverband in der deutschen Strom- und Energiebranche. Dieser vertritt über 1.800 Unternehmen im Kampf gegen beispielsweise staatliche Vorgaben und Forcierung des Wettbewerbs, um die Gewinne der Unternehmen zu erhalten oder zu steigern.

Business Use Case (BUC)

Der Business Use Case (*dt.* Geschäftsanwendungsfall) beschreibt den übergeordneten Use Case im Smart Grid Architecture Model-Modell. Dazu werden die alle beteilligten Akteure und ihre Business Goals (*dt.* Geschäftsziele) dargestellt und beschrieben.

Capability Maturity Model Integration

Bei Capability Maturity Model Integration (CMMI) handelt es sich eine Familie von Referenzmodellen für das Qualitätsmanagement. Diese enthält auch spezielle Modelle für das Qualitätsmanagement in Projekten.

Common Information Model

Das Common Information Model ist ein Informationsmodell zur architekturübergreifenden Systemmodellierung.

Confluence

Confluence ist eine kommerzielle Wiki-Software, die vom australischen Unternehmen Atlassian entwickelt und als Enterprise Wiki hauptsächlich für die Kommunikation und den Wissensaustausch in Unternehmen und Organisationen verwendet wird (siehe Unterunterabschnitt 2.5.4).

Constraint

(Technische) Einschränkung bezogen auf die Möglichkeiten eines Fahrplans bzw. einer Anlage.

CRUD

Die Abkürzung CRUD steht für die Datenbankoperationen create, read, update und delete.

Cyber-physikalische Systeme

Cyber-physikalische Systeme sind Systeme, die aus einem Verbund informatischer, softwaretechnischer Komponenten und mechanischen und elektronischen Teilen bestehen.

Day-Ahead-Handel

Unter dem Begriff Day-Ahead-Handel versteht man den Handel von Strom für den folgenden Tag, der an der EPEX SPOT in Paris (Spotmarkt der European Power Exchange), an der EXAA in Wien (Energy Exchange Austria) oder im OTC (Over-the-Counter-Handel) über außerbörslich ausgehandelte Verträge stattfindet. Stellenweise wird auch der Begriff Auktionsmarkt verwendet.

Direktvermarkter

Als Direktvermarkter wird jemand bezeichnet, der das Portfolio an die Strombörse verkauft.

Erneuerbare-Energien-Gesetz

Deutsches Gesetz, das die Einspeisung von Strom aus erneuerbaren Quellen bevorzugt und feste Einspeisevergütung garantiert. Der komplette Titel des Gesetzes lautet *Gesetz für den Ausbau erneuerbarer Energien*.

Etherpad

Das Etherpad ist eine freie Server-Software, welche einen webbasierten Texteditor bereitstellt. Mit diesem kann ein Dokument gleichzeitig von vielen Personen bearbeitet werden.

European Power Exchange Spotmarket

Tagvorausbörse für Strom, bei der mit verschiedenen Paketen gehandelt wird. Die angebotenen Pakete umfassen jeweils eine feste Lieferung von mindestens 0,1 MW für einen bestimmten Zeitraum, beispielsweise *High Noon* von 11 Uhr mittags bis 14 Uhr nachmittags.

Experiment

Als Experiment wird ein einzelner Durchlauf der Evaluation bezeichnet. Dabei wird die Optimierung für ein konkretes Szenario durchgeführt (s. Abschnitt 6.10.1).

Flexibilitäten

Menge von gültigen (im Rahmen der technischen Möglichkeiten durchführbaren) Fahrplänen einer Anlage (inklusive der entstehenden Kosten).

Git

Git ist eine Software die zur verteilten Versionierung von Dateien eingesetzt werden kann. Siehe dazu auch Kapitel 2.5.1

grafische Benutzeroberfläche

Eine grafische Benutzeroberfläche (GUI, engl. graphical user interface) bezeichnet eine grafische Benutzerschnittstelle.

Haushaltstyp

Ein Haushaltstyp beinhaltet ein Standardlastprofil für einen Verbraucher. Das Lastprofil beschreibt dabei, um welche Art von Verbraucher es sich handelt, bspw. Gewerbe oder Haushalt, und gibt an, zu welchem Zeitpunkt, wie viel Wärmeenergie benötigt wird. Die Standardlastprofile werden vom BDEW (Bundesverband der Energie- und Wasserwirtschaft e.V.) erstellt und auch von den Energielieferanten zu Abrechnungs- und Kalkulationszwecken verwendet.

Heuristik

Im Operations Research versteht man unter Heuristiken Verfahren zur Lösung komplexer Optimierungsprobleme. Hierbei wird die gegebene Problemstruktur in geeignet erscheinender Weise berücksichtigt, um entweder möglichst schnell zu einer akzeptablen Lösung oder in akzeptabler Zeit zu einer möglichst guten Lösung zu gelangen.

Hibernate

Hibernate ist ein Framework, mit dessen Hilfe sich Java-Objekte leicht in eine relationale Datenbank speichern lassen.

High Level Use Case

Ein Anwendungsfall, der die allgemeine Idee einer Funktion beschreiben soll (und ohne Vorwissen in Informatik verstanden werden kann).

Imputation

Die Imputation gehört zu den sogenannten *Missing-Data-Techniken*, also Verfahren, die bei Auswertung unvollständiger Stichprobendatensätze angewendet werden.

innerer Optimierer

Jeder innere Optimierer gehört zu einem äußeren Optimierer und wird von diesem aufgerufen. Im Rahmen der Projektgruppe wurden die Produktportfoliooptimierer (Linear-Programming-Optimierer und Rekursiv-Intervall-Optimierer) als innere Optimierer implementiert (mehr dazu in Abschnitt 6.9.2).

Intraday-Handel

Kurzfristiger Stromgroßhandel an der EPEX SPOT (Spotmarkt der European Power Exchange). Er bezeichnet den kontinuierlichen Kauf und Verkauf von Strom, der noch am gleichen Tag geliefert wird.

ISO 10006

ISO 10006 basiert auf den Richtlinien der ISO 9001, ist aber speziell für das Qualitätsmanagement von Projekten entwickelt worden.

ISO 9001

Bei der ISO 9001 handelt es sich um eine Qualitätsmanagementnorm, nach der Unternehmen sich zertifizieren lassen können. Diese beinhaltet Richtlinien, wie das Qualitätsmanagement umgesetzt werden soll.

JUnit

JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist.

K-nearest neighbors

Als Kreuzvalidierung wird ein Verfahren für maschinellem Lernen bezeichnet. In diesem Verfahren wird eine Teilmenge der Daten dazu genutzt, um das erlernte Modell mit wirklichen Werten zu vergleichen und eine Güte zu erstellen. Diese Güte kann genutzt werden, um verschiedene Parameterkombinationen zu vergleichen und die beste auszuwählen.

K-nearest neighbors

K-nearest neighbors (dt. Nächste Nachbarn) ist ein Algorithmus zur Klassifikation und Regression von Daten.

kontinuierliche Integration

Kontinuierliche Integration (CI, engl. continuous integration) ist ein Begriff aus der Software-Entwicklung, der den Prozess des fortlaufenden Zusammenfügens von Komponenten zu einer Anwendung beschreibt.

Kraft-Wärme-Kopplung

Kraft-Wärme-Kopplung bezeichnet ein Energieerzeugungsprinzip, bei dem gleichzeitig elektrische und thermische Energie erzeugt und zur Nutzung bereitgestellt werden.

Leistungsverlauf

Zeitlicher Verlauf von elektrischer Leistung, i.d.R. stündlich oder viertelstündlich aufgelöst.

Log4j

Log4j ist ein Logging Framework von Apache.

Lombok

Lombok ist eine IDE-Erweiterung, welche spezielle Annotationen im Java-Code erkennt und kompiliert. Auf diese Weise werden Getter und Setter automatisch generiert.

LVK

Name des Loggers, der von der Projektgruppe auf Basis von Log4j entwickelt wurde.

Marktdatenprovider

Stellt historische oder aktuelle Marktdaten der Strombörse bereit.

Maven

Maven ist ein Framework, welches die Softwareentwicklung beim Kompilieren, Testen und Packen unterstützt.

Mosaik

Mosaik ist ein flexibles Co-Simulationsframework, speziell auf die Simulation von Energienetzen und Energiekomponenten ausgelegt.

OPC Unified Architecture

Das Open Platform Communications Unified Architecture ist ein Kommmunikationsprotokoll. Es wurde im Rahmen der Projektgruppe zur Kommunikation zwischen *Powder* und potentiellen realen Anlagen wie beispielsweise BHKW oder PV-Anlagen.

Operations Research

Unter dem Begriff Operations Research versteht man die Entwicklung und den Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung in Unternehmen und Organisationen.

parametrisierter Anlagentyp

Ein parametrisierter Anlagentyp bezeichnet einen Anlagentyp, der eine bestimmte Leistung liefert. Beispielsweise bezeichnet der parametrisierte Anlagentyp "PV_10KW" eine Anlage vom Typ PV, deren Maximalleistung 10 Kilowatt beträgt.

Photovoltaik-Anlage

Eine Photovoltaikanlage ist eine Anlage zur Erzeugung von elektrischer Energie. Dabei wird die Einstrahlung der Sonne in den Kollektoren aufgenommen und in elektrische Energie umgewandelt. Photovoltaikanlagen gehören zu den Anlagen zur Erzeugung von erneuerbarer Energie.

Primary Use Case

Anwendungsfall, der näher auf die Implementierung eingeht und den HLUC verfeinert.

Programmierschnittstelle

API (engl. application programming interface). Programmteil der anderen Programmen als Schnittstelle zur Verfügung gestellt wird.

Pufferspeicher

Ein Pufferspeicher speichert thermische Leistung in Form von temperiertem Wasser. Ist ein BHKW in Betrieb, speist es die erzeugte thermische Leistung in den Pufferspeicher ein, dessen Temperatur sich infolgedessen erhöht. Verbraucht der Haushalt warmes Wasser (z. B. für die Heizung), verringert sich die Temperatur.

PV

PV-Anlage ist die Abkürzung von Photovoltaikanlage. Bei dieser handelt es sich um eine Anlage zur Erzeugung von elektrischer Energie. Dabei wird die Einstrahlung der Sonne in den Kollektoren aufgenommen und in elektrische Energie umgewandelt. Photovoltaikanlagen gehören zu den Anlagen zur Erzeugung von erneuerbarer Energie.

reale Anlage

Als reale Anlage wird eine Anlage bezeichnet, die aus physischen Komponenten besteht. Dieser Begriff dient zur Abgrenzung zu virtuellen Anlagemodellen.

Remote Method Invocation

Java-eigene Implementierung des Remote Procedure Calls.

Remote Procedure Call

Aufrufen einer Prozedur auf einem entfernten System.

Repositorium

Ein Git Repositorium ist ein Verzeichnis in dem Dateien inklusive einer Versionsgeschichte verwaltet werden.

Sampling

Das Sampling ist eine Methode, bei der aus einem gegebenen Datenraum beliebiger Dimensionalität eine Menge an Datenpunkten (sampling points) ausgewählt wird. Die Gesamtheit der ausgewählten Datenpunkte bildet ein Sample.

Scrum Unified Process

Das von der Projektgruppe für dieses Projekt entwickelte Vorgehensmodell, welches Elemente von Scrum und Unified Process miteinander kombiniert.

Slack

Slack ist eine Software, welche zur Kommunikation innerhalb von Arbeitsgruppen genutzt werden kann.

Smart Grid Architecture Model

Referenzmodell zur Analyse und Visualisierung von intelligenten Stromnetzen.

Smart Grid Plane

Die Smart Grid Plane ist Teil jeder Ebene des SGAMs. Sie besteht aus den zwei Dimensionen der Zonen (hierarische Ebenen) und Domainen (Energiekonvertierungskette) .

Support Vector Machine

Die SVM (Support Vector Machine) ist ein Algorithmus zur Klassifikation und Regression von Daten.

Szenario

Ein Szenario stellt eine konkrete Konfiguration eines VK dar, wie beispielsweise die Gesamtleistung des VK, die registrierten Anlagen und die Anlagenflexibilitäten. Szenarien werden für die Evaluation der Optimierer verwendet: In jedem Experiment wird ein Szenario ausgeführt (s. Abschnitte 6.1 und 6.10.1).

Time Series Prediction

Bei Time Series Prediction (dt. Zeitreihenvorhersage oder -prognose) handelt es sich um eine Art von Vorhersageverfahen. Mit ihr werden aus Vergangenheitsdaten Prognosen erstellt und dabei die Zeit als Reihenfolge beachtet.

Unified Process

Vorgehensmodell des Projektmanagements, welches über die Umsetzung sechs ausgewählter wichtiger Aspekte den Prozess der Entwicklung steuert. Die Aspekte sind iterative Softwareentwicklung, Anforderungsmanagement, Änderungsmanagement, komponentenbasierte Architektur, visuelle Modellierung und kontinuierliche Qualitätsprüfung. Näheres zur spezifischen Struktur kann im Buch von Essigkrug und Mey [EM03] nachgeschlagen werden.

Unteranpassung

Als Unteranpassung wird Modell genannt, dass zu allgemeingültig ist. Es gilt als Gegenteil von Überanpassung.

virtuelles Kraftwerk

Ein Virtuelles Kraftwerk aggregiert die Leistungen all der ihm zugehörigen Anlagen, so dass die Leistung aus verschiedenen Anlagen gebündelt bereitgestellt werden kann. Die dezentralen Energieerzeugungsanlagen werden durch eine zentrale Steuerung koordiniert, zusammengefasst und bilden so von außen betrachtet ein einzelnes Kraftwerk. Andere Namen für virtuelle Kraftwerke sind beispielsweise Schwarmkraftwerk, Kombikraftwerk oder Hybridkraftwerk.

VK-Betreiber

Der VK-Betreiber ist der Besitzer und Leiter eines virtuelles Kraftwerk. Ihm obliegt die Wartung und der Einstatz der verwendeten Software des VKs, sowie die Kommunikation mit den Anlagenbetreibern und ggf. dem Handel an der Börse. Auch übernimmt dieser die Aufnahme von neuen Kraftwerken, über ihre Stammdaten, im virtuellen Kraftwerk.

Waikato Environment for Knowledge Analysis

Weka ist ein Softwaretool, das verschiedene Techniken aus den Bereichen Maschinelles Lernen und Data-Mining bereitstellt. Es wurde an der University of Waikato entwickelt und ist in Java geschrieben. Es handelt sich um eine frei verfügbare Software, die unter der GNU General Public License steht.

Weather Information Exchange Model

Datenmodellstandard für Wetterdaten.

Web Scraping

Web Scraping ist eine Technik, die dazu dient, Informationen aus Webseiten zu extrahieren und in eine strukturierte Form umzuwandeln.

Wetterdatenprovider

Stellt historische oder aktuelle Wetterdaten bereit.

Überanpassung

Als Überanpassung wird eine zu starke Anpassung des Modells an eine bestimmte Datenmenge bezeichnet. Für andere ähnliche Datenmengen kann das Modell nicht korrekt funktionieren.

äußerer Optimierer

Als äußerer Optimierer wird der bezeichnet, mit dem die Optimierung begonnen wird. Das kann ein Produktportfolio- oder ein Einsatzplanoptimierer sein. Jeder äußere Optimierer besitzt einen inneren Optimierer und ruft diesen während der Optimierung mehrmals auf. Im Rahmen der Projektgruppe wurden die Einsatzplanoptimierer (Simulated Annealing- und Tabusuche-Optimierer) als äußere Optimierer implementiert (mehr dazu in Abschnitt 6.9.3).

Stichwortverzeichnis

T