



**mosaik Advanced Visualization  
Environment for Intelligent Power Grids**

# Projektdokumentation

---

13. März 2015

Erika Root, Gerrit Klasen, Hanno Günther,  
Jerome Tammen, Marina Sartison, Marius Brinkmann,  
Michael Falk, Rafael Burschik, Rouven Pajewski  
Sascha Spengler, Andrianarisoa A. Johary Ny Aina und  
Tobias Schwerdtfeger



E-Mail: [pg-maverig@offis.de](mailto:pg-maverig@offis.de)

Maverig ist eine Benutzungsoberfläche zum Erstellen und Visualisieren von Smart Grid Simulationen.

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>5</b>
1.1. Motivation der Projektgruppe . . . . .	5
1.2. Zielsetzung der Projektgruppe . . . . .	5
1.3. Aufbau der Dokumentation . . . . .	6
<b>2. Projektorganisation</b>	<b>7</b>
2.1. Projektinterne Aufgabenbereiche . . . . .	7
2.1.1. Dokumenten-Manager . . . . .	7
2.1.2. GUI-Beauftragter . . . . .	7
2.1.3. Konfigurationsmanager . . . . .	8
2.1.4. Product Owner . . . . .	8
2.1.5. Scrum Master . . . . .	9
2.1.6. Testmanager . . . . .	9
2.2. Vorgehensweise . . . . .	10
2.2.1. Scrum . . . . .	11
2.2.2. Testmanagement . . . . .	11
2.3. Angewandte Software . . . . .	14
2.3.1. Bitbucket . . . . .	14
2.3.2. Confluence . . . . .	15
2.3.3. Deployment-Tools . . . . .	15
2.3.4. Force Atlas 2 . . . . .	16
2.3.5. JIRA . . . . .	17
2.3.6. L <sup>A</sup> T <sub>E</sub> X . . . . .	18
2.3.7. Mercurial . . . . .	19
2.3.8. Mosaik . . . . .	19
2.3.9. Poedit . . . . .	21
2.3.10. PyCharm . . . . .	21
2.3.11. PySide . . . . .	21
2.4. Softwarelizenzierung . . . . .	22
2.4.1. Vorgehensweise für die Softwarelizenzierung . . . . .	22
2.4.2. Wahl der Software-Lizenz . . . . .	22



<b>3. Grundlagen</b>	<b>24</b>
3.1. Arbeit von Energiesystemen . . . . .	24
3.1.1. Physikalische Grundbegriffe . . . . .	24
3.1.2. Erzeugung elektrischer Energie . . . . .	25
3.1.3. Übertragung, Transport und Verteilung von elektrischer Energie	29
3.1.4. Netztopologien . . . . .	31
3.1.5. Systemdienstleistungen . . . . .	34
3.2. Simulation von Smart Grids . . . . .	35
3.2.1. Begriffliche Abgrenzungen . . . . .	36
3.2.2. Systemarchitektur . . . . .	37
3.2.3. Kommunikation . . . . .	39
3.2.4. Herausforderungen . . . . .	40
3.2.5. Ausblick . . . . .	42
<b>4. Anforderungen</b>	<b>44</b>
4.1. Funktionale Anforderungen . . . . .	44
4.2. Nichtfunktionale Anforderungen . . . . .	51
<b>5. Systementwurf</b>	<b>52</b>
5.1. Zweck des Systems . . . . .	52
5.2. Anwendungsfalldiagramm . . . . .	52
5.3. Entwurfsziele . . . . .	53
5.4. Einsatz von 2D und Einblick in 3D . . . . .	54
5.5. Zerlegung des Systems: Model-View-Presenter . . . . .	55
5.6. Mock-Ups . . . . .	57
<b>6. Implementierung</b>	<b>65</b>
6.1. Model . . . . .	66
6.2. View . . . . .	70
6.3. Presenter . . . . .	75
6.4. Data . . . . .	76
6.5. Util . . . . .	83
6.6. Anwendungsfall . . . . .	87
<b>7. Tests und Performance</b>	<b>89</b>
7.1. Testdokumentation . . . . .	89
7.2. Usability-Tests . . . . .	94
7.2.1. Usability-Studie I . . . . .	95
7.2.2. Usability-Studie II . . . . .	97
7.3. Performance . . . . .	100
7.3.1. Evaluation von ForceAtlas 2 . . . . .	100



7.3.2.	Evaluation der Element-Selektion . . . . .	103
<b>8.</b>	<b>Projektabschluss</b>	<b>105</b>
8.1.	Fazit . . . . .	105
8.2.	Ausblick . . . . .	106
8.2.1.	Konzept für mehrfaches Erstellen und Verbinden von Elementen . . . . .	107
8.2.2.	Konzept Heat-Werte der einzelnen Elemente . . . . .	108
8.2.3.	Konzept für Integration von Kontrollmechanismen . . . . .	109
8.2.4.	Weiterentwicklungsmöglichkeiten für Maverig . . . . .	112
	<b>Literaturverzeichnis</b>	<b>114</b>
<b>A.</b>	<b>Anhang</b>	<b>118</b>
A.1.	Benutzerhandbuch . . . . .	118
A.2.	User Manual . . . . .	136
A.3.	Funktionsumfang . . . . .	154

# 1. Einleitung

Im Rahmen der Masterstudiengänge Wirtschaftsinformatik und Informatik an der Carl von Ossietzky Universität Oldenburg ist für jeden Studierenden die Vorgabe, an einer einjährigen Projektgruppe teilzunehmen und eine vorgegebene Aufgabenstellung zu bearbeiten. Die Projektgruppe *Maverig II* wurde von Prof. Dr. Sebastian Lehnhoff im *OFFIS Institut* initiiert. Der Start der Projektgruppe war der 01. April 2014 und das Ende nach einem Jahr der 31. März 2015. Die Projektgruppe besteht aus zwölf Studierenden, die verschiedene Rollen, wie in Abschnitt 2.1 zugeordnet, wahrnehmen. Zu Beginn des Projekts war eine theoretische Phase vorgesehen, in der jeder Projektteilnehmer sich eingängig mit einem projektbezogenen Thema beschäftigt. Hierbei wurde eine Ausarbeitung wissenschaftlich erarbeitet und in zwei Blockseminaren vorgetragen.

## 1.1. Motivation der Projektgruppe

Eine aktuelle Herausforderung für die Energiebranche in Deutschland ist die Energiewende. Einen Teil dazu trägt das Themengebiet der intelligenten Energienetze, so genannter Smart Grids, bei. Um die Kompliziertheit derartiger Technologie besser händeln zu können, wurde das Simulationssystem *Mosaik* am OFFIS Institut entwickelt. Es existiert bisher keine intuitive und umfassende Oberfläche, um ein Smart Grid Szenario für die Lehre oder im praktischen Umfeld mit *Mosaik* erstellen zu können. Die mögliche virtuelle Simulation eines erstellten Szenarios liefert ebenfalls einen wesentlichen Nutzen für den effektiven und effizienten Einsatz von Smart Grid Technologie in Forschung und Wirtschaft.

## 1.2. Zielsetzung der Projektgruppe

Das Ziel der Projektgruppe ist die Neuentwicklung einer Benutzungsoberfläche zum Erstellen und Visualisieren von Smart Grid Simulationen, die *Maverig* genannt wird. Die Oberfläche soll zum einen dazu dienen, ein Smart Grid Szenario mit diversen Stromnetzteilnehmern in Form von verbundenen Verbrauchern und Erzeugern zu konzipieren. Zum anderen soll eine zeitliche Simulation des zuvor erstellten Stromnetzes durchgeführt werden können. Dafür sind zunächst Anforderungen in Zusam-

menarbeit mit dem Auftraggeber zu ermitteln und zu dokumentieren. Der Auftraggeber des Projekts ist die Abteilung Energieinformatik der Universität Oldenburg unter der Leitung von Prof. Dr. Sebastian Lehnhoff. Die Abteilung Energieinformatik beschäftigt sich in verschiedenen Forschungsprojekten mit den Problemen der Energiewende, insbesondere Smart Grid Netzen. Für den Ablauf der Smart Grid Simulation soll das bereits existierende Framework *Mosaik*, das in Kapitel 2.3.8 näher erläutert wird, verwendet werden. Dafür muss eine Schnittstelle zwischen der zu entwickelnden Desktopanwendung und dem *Mosaik* Framework erstellt werden. Um sicherzustellen, dass die entwickelte Oberfläche die Anforderungen korrekt umsetzt, muss in der Entwicklung regelmäßig eine Evaluation stattfinden.

### 1.3. Aufbau der Dokumentation

Die Dokumentation des Projekts gliedert sich in einzelne Abschnitte auf, die im Folgenden näher beschrieben werden sollen. In Kapitel 2 geht es um die organisatorischen Aspekte des Projekts. Neben den projektinternen Aufgabenbereichen wird die Vorgehensweise beschrieben, die in diesem Projekt verwendet wird. Weiterhin werden in diesem Abschnitt die eingesetzte Software und die Softwarelizenzierung betrachtet. Im nächsten Kapitel (siehe Kapitel 3) werden die benötigten Grundlagen, insbesondere das Themenfeld *Dezentrale Energiesysteme* betreffend, aufgeführt. Nach dem Grundlagenteil werden in Kapitel 4 die funktionalen und nicht-funktionalen Anforderungen aufgelistet. Der Systementwurf in Kapitel 5 beschreibt die konzeptionelle Sicht auf die Software. Außerdem sind in diesem Kapitel Designentscheidungen, die die Architektur des Systems betreffen, dokumentiert sowie entwickelte Mock-Ups hinterlegt. Nach dem Systementwurf folgt das Kapitel 6 mit der Dokumentation der Implementierung, wobei auf die Projektstruktur und ihren Inhalt näher eingegangen wird. Anschließend ist im Kapitel 7 die Testdokumentation sowie die Dokumentation der zwei durchgeführten Usability Studien vorzuführen. Im letzten Kapitel wird das Projekt dann hinsichtlich des Projekterfolges abschließend betrachtet und ein Ausblick für weitere Konzepte, die die Funktionalität von *Maverig* erweitern, aufgeführt.

## 2. Projektorganisation

Im folgenden Kapitel werden die organisatorischen Aspekte des Projektes dargestellt. Dies beinhaltet die projektinternen Aufgabenbereiche sowie die Vorgehensweise während des Projektes. Des Weiteren werden die im Projekt angewandte Software aufgeführt.

### 2.1. Projektinterne Aufgabenbereiche

In unserem Projekt wurde jedem Teammitglied eine Rolle zugewiesen, die er durchgehend wahrgenommen hat. Nachfolgend werden die einzelnen Rollen und ihre Aufgaben näher erläutert und die verantwortlichen Mitglieder zugewiesen.

#### 2.1.1. Dokumenten-Manager

Der *Dokumenten-Manager* ist für die Erstellung und Pflege aller projektrelevanten Dokumente zuständig. Hierzu zählen die Initial-Dokumentation, Software-Dokumentation, Benutzerhandbuch oder aber auch Confluence Templates, sowie wöchentlich generierte Dokumente wie Protokolle. Alle anfallenden Dokumente werden vom Dokumenten-Manager gesichtet und archiviert. Er ist nicht ausschließlich alleine für sämtliche Inhalte zuständig, diese werden im Kollektiv zusammengetragen. Der Dokumenten-Manager ist für die Integration aller Inhalte in das Abschlussdokument zuständig, damit diese strukturiert und übersichtlich zur Verfügung stehen. Als Grundlage für die Erstellung von Berichten und Protokollen wird LaTeX verwendet. Die Aufgabe des Dokumenten-Managers wird von Rouven Pajewski und Gerrit Klasen übernommen.

#### 2.1.2. GUI-Beauftragter

Die Rolle des *GUI-Beauftragten*, wahrgenommen durch Tobias Schwerdtfeger und Jerome Tammen, ist für die allgemeine Festlegung verschiedener Design-Grundlagen und deren Einhaltung verantwortlich. Dabei wird besonders Wert auf eine konsistente Darstellung der zu erstellenden Software geachtet. Hierbei stehen Benutzerfreundlichkeit und Usability der grafischen Oberfläche (GUI) an erster Stelle. Zusätzlich sollen Effektivität und Effizienz der GUI auf einem hohen Level gehalten

werden. Fehlertoleranz (rate of errors by users), Aufgabenangemessenheit, Steuerbarkeit und Individualisierbarkeit sind weitere Faktoren, die durch die Rolle des GUI-Beauftragten im Laufe des Projektes immer wieder kritisch beleuchtet werden sollen.

Organisatorisch wird er der Ansprechpartner für die Projektgruppe sein, der bei Designfragen unterstützt und Verantwortung übernimmt. Somit liegt bei ihm auch die letzte Kontrolle der Benutzeroberfläche, die dem Anwender präsentiert wird. Entscheidungen können jedoch nicht eigenmächtig getroffen werden und sind stets in Rücksprache mit der Gruppe zu beschließen. Sollte es Benutzerstudien im Hinblick auf Design-Entscheidungen geben, sind diese vom GUI-Beauftragten zu verwalten.

### 2.1.3. Konfigurationsmanager

Die Rolle des *Konfigurationsmanagers* nehmen Michael Falk und Sascha Spengler wahr. Der Konfigurationsmanager kümmert sich um die Bereitstellung benötigter Infrastrukturen zur Entwicklung einer Software. Dazu zählt in der Projektgruppe vor allem die Bereitstellung mehrerer Mercurial-Repository zur Versionsverwaltung von Software und Dokumentation sowie von Werkzeugen für das Projektmanagement. In der Projektgruppe werden für das Projektmanagement Jira und Confluence von Michael und mehrere Repository über Bitbucket von Sascha bereitgestellt, konfiguriert und verwaltet. Zusätzlich überwacht das Konfigurationsmanagement die Deployment-Tätigkeiten und führt für das Deployment die Paketerstellung sowie Konvertierung der Software selbst durch. Des Weiteren sind die Konfigurationsmanager Ansprechpartner für die anderen Projektmitglieder bei Problemen mit den im Projekt genutzten Werkzeugen. Neben dem Konfigurationsmanagement werden von Michael und Sascha auch Entwicklertätigkeiten wahrgenommen. Konfigurationsmanager kümmern sich nicht um die Einrichtung aller Werkzeuge auf den Systemen der Projektmitglieder.

### 2.1.4. Product Owner

Die Rolle der *Product Owner* nehmen Johary Andrianarisoa und Hanno Günther wahr. Die Product Owner entscheiden, welche Features in welcher Form für die Zielerfüllung des Projektes relevant sind. Sie sollen eine klare Vision für das auszuliefernde Produkt vorgeben, damit die Scrum-Master mit dem Team auf dieser Basis konkrete Ziele erfassen können. Die Funktionalitäten sollen daraufhin in den jeweiligen Sprints möglichst effektiv umgesetzt werden. Die Product Owner erstellen und pflegen das Product Backlog inkl. der User Stories, welche in regelmäßigen Abständen im Scrum-Team besprochen werden, um daraus einzelne Aufgaben für das Sprint Backlog abzuleiten. Während der Projektlaufzeit kommunizieren die Product



Owner mit den Auftraggebern und sorgen dafür, dass deren Ziele und Anforderungen möglichst gut eingehalten werden. Sie sind somit gleichzeitig auch Ansprechpartner für Rückfragen bezüglich der geplanten Umsetzung. In dieser Projektgruppe werden die Product Owner auch am Entwicklungsprozess teilnehmen. Die Aufgabe von Product Owner ist nicht die Erstellung und Aufwandsabschätzung von konkreten Aufgaben. Die Product Owner müssen in erster Linie die Kundenerwartungen berücksichtigen und hierbei Priorisierungskonflikte bei der Umsetzung lösen.

### 2.1.5. Scrum Master

Die Rolle des *Scrum Master* nehmen Marius Brinkmann und Rafael Burschik wahr. Dabei sind sie in der Projektgruppe zuständig, den Entwicklungsprozess gemäß Scrum einzuhalten. Dazu führen sie die Scrum-Regeln ein, überprüfen deren Einhaltung und versuchen jede Störung des Scrum-Prozesses zu beheben. Demzufolge ist der Scrum Master verantwortlich für die Beseitigung von Hindernissen, wie Problemen im Entwicklungsteam oder Scrum-Team, sowie Störungen von außen.

Des Weiteren gibt es kein separates Projektmanagement. Die Projektmanagement-Tätigkeiten, wie z.B. die zeitliche Planung, der Soll/Ist-Vergleich und die Meilensteinplanung gehören somit zu den Aktivitäten des Scrum Masters. In jeder Sitzung wird zu Beginn ein *Weekly Scrum* durchgeführt, welches abwechselnd von Marius und Rafael moderiert wird. Außerdem ist die Pflege des *Sprint Backlog*, *Impediment Backlog* und der *Definition of Done* ein Bestandteil der Rolle des Scrum Masters. Das *Product Backlog* gehört nicht zu dem Tätigkeitsbereich des Scrum Masters, sondern wird durch die *Product Owner* geführt. Entgegen der konventionellen Aufteilung in Scrum, werden die Scrum Master in der Projektgruppe auch Entwicklungstätigkeiten wahrnehmen, sofern der Koordinationsprozess davon nicht beeinträchtigt wird.

### 2.1.6. Testmanager

Die Rolle des *Testmanagers* wird von Erika Root und Marina Sartison wahrgenommen. Der Testmanager beschäftigt sich mit der Definition, Koordinierung und Kontrolle der Testaktivitäten. Des Weiteren sind die Testmanager für die Planung und Umsetzung der Testumgebung zuständig. Zu dem Aufgabengebiet eines Testmanagers gehört NICHT die Erstellung, Durchführung und Dokumentation der Testfälle.

#### 1. Definition:

- *Definition of Test* beinhaltet die Vorgaben bezüglich der Qualitätssicherung und Test. Aspekte, die zu berücksichtigen sind, sind unter anderem die Art der Dokumentation der Testfälle und deren Ergebnisse, welche Testmethoden und -werkzeuge eingesetzt werden, sowie die der Grad der Testabdeckung.

- *Definition of Ready* definiert Kriterien bezüglich der Umsetzung der User Stories. Dies beinhaltet auch die Beantwortung der Fragen: Warum? Was? Wie? Klar definiert? Handhabbar? Testbar?
- *Definition of Done* hält aus Sicht des Teams fest, welche Kriterien erfüllt sein müssen, damit eine User Story am Sprintende als „fertig“ angesehen wird und dem Product Owner übergeben werden kann.

2. Koordinierung und Kontrolle

- Ziel ist den Grad der Testabdeckung zu erreichen.
- Die Einhaltung aller zuvor definierter Dokumente werden überprüft.
- Nach jedem Sprint wird das Dokument „Definition of Test“ überprüfen und falls Änderungen in den Anforderungen oder Risiken des Produktes bestehen ggf. aktualisiert.

## 2.2. Vorgehensweise

Wir haben zu Beginn den Projektablauf mit Hilfe einer Roadmap geplant und visualisiert. Hierdurch war es möglich, kontinuierlich den Soll/Ist-Vergleich und Verzüge abschätzen zu können.

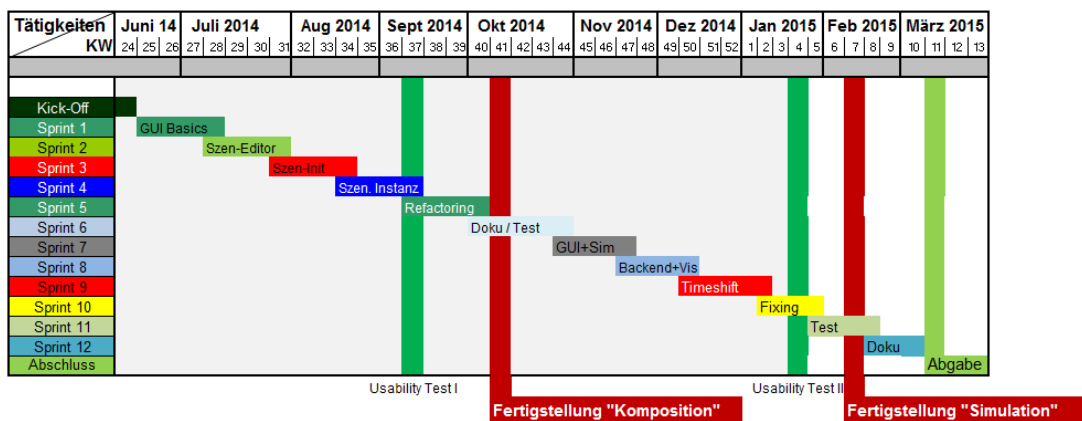


Abbildung 2.1.: Roadmap Maverig

Wie in der Abbildung 2.1 zu erkennen, sind 12 Sprints mit jeweils einer Länge zwischen drei und vier Wochen geplant. Das Gesamtprodukt wurde in zwei Arbeitspakete geteilt, der *Komposition* und der *Simulation*. In jedem Sprint sollte eine umfangreiche Funktion realisiert werden, um bei jedem Sprintende ein auslieferbares Produkt zu erhalten. Neben den Meilensteinen sind zwei Usability-Tests vor jeder

Fertigstellung geplant, um eine größtmögliche Benutzbarkeit und Gebrauchstauglichkeit zu gewährleisten. Des Weiteren werden auch Sprints exklusiv für das Testen und Dokumentieren durchgeführt, um eine hohe Qualität des Projekts zu gewährleisten.

### 2.2.1. Scrum

Als relevantes Vorgehensmodell für die Projektgruppe wurde *Scrum* ausgewählt, da es großen Wert auf die Entwicklung eines Produktes im Team legt. Nach Betrachtung unterschiedlicher Vorgehensmodelle stellt sich Scrum als für die Projektgruppe optimalste Lösung dar. Aufgrund der bisher gesammelten Erfahrungen im Softwareprojekt, ist einem Großteil der Mitglieder der Scrum-Prozess und dessen Umsetzung bekannt. Dazu kommen die in Scrum definierten Iterationen, die eine regelmäßige Anpassung der Anforderungen sowie des zu entwickelnden Produktes ermöglichen und somit keine detaillierte Planungsphase zu Beginn des Projektes erfordern.

Die Scrum unterstützende Software ist aus dem Angebot des Softwareherstellers *Atlassian* ausgewählt. Auf der einen Seite dient *Jira* als Projektmanagement-Tool und auf der anderen Seite wird *Confluence* als kollaborative Textbearbeitungs-Software genutzt.

### 2.2.2. Testmanagement

Das Testmanagement ist ein wichtiger Bestandteil jeder Softwareentwicklung. Dies soll sicherstellen, dass das Endprodukt keine Mängel in der Funktionalität aufweist. Nachfolgend wird demnach festgelegt, wie wir beim Testen der Software vorgehen und welche Kriterien dabei zu beachten sind.

#### Definition of Test

Dieses Dokument beinhalten die Vorgaben der Dokumentation der Testumgebung und die Vorgehensweise der Testaktivitäten.

1. Innerhalb der Entwicklungsumgebung *PyCharm* wird mit dem Testwerkzeug *Pytest* getestet. *Pytest* ermöglicht es Unittests zu starten. In den Unittest werden innerhalb von einzelnen Testfällen Teilfunktionalitäten des Codes auf die gewünschte Funktionalität überprüft.
2. Die Dokumentation der Testfälle ist wie folgt geregelt:
  - a) Die Testfälle müssen alle im Programmcode durch Kommentare erklärt werden. Dies enthält welche Teilfunktionalität durch den entsprechenden Testfall überprüft wird.

- b) Die Dokumentation der Unittests, Integrationstests, Regressionstests, Systemtests und Sprachtests müssen in einem entsprechenden Kapitel mit einem entgeltigen Ergebnis und Verhalten in einem dafür vorgesehenen Sprint festgehalten werden.
- Unittest: Testen der einzelnen Teilfunktionalitäten (Methoden/Funktionen)
  - Integrationstest: Testen der Zusammenwirkung von mehreren Komponenten
  - Regressionstest: Wiederholung von Testfällen
  - Systemtest: Überprüft das Gesamtsystem unter möglichst realistischen Bedingungen
  - Sprachtest: Überprüfung einer sinngemäßen Bedeutung der unterschiedlich angebotenen Sprachen der Desktopanwendung

3. Es ist wie folgt beim Testen vorzugehen:

- Es ist eine siebzig-prozentige Testabdeckung zu erreichen, d.h. die Teilfunktionalitäten (Funktionen) müssen durch mindestens einen Testfall überprüft werden.
- Es soll überprüft werden, ob alle Anforderungen im Sprint entsprechend umgesetzt wurden.
- Nach der Implementierung einer Teilfunktion soll diese anschließend zeitig innerhalb eines dafür vorgesehenen Sprints getestet werden.

### Definition of Ready

Dieses Dokument beschreibt, wann eine User Story als „bereit“ zur Entwicklung klassifiziert wird und auch am Ende des Sprints als „done“ klassifiziert werden kann. Sind User Stories (US) „ready“ werden sie aus dem Product Backlog ins Sprint Backlog aufgenommen. Das Dokument soll als eine Art Kriterienkatalog genutzt werden.

- Warum soll die US umgesetzt werden? Welches Problem wird damit behoben?
- Die Details sind für alle ausreichend verständlich?
  - Kann eine Entscheidung getroffen werden, ob die Fertigstellung einer User Story bezüglich ihrer Komplexität und Größe möglich ist?
  - Ist der Aufwand der Fertigstellung bekannt?
  - Wie soll die US umgesetzt werden?
  - Was ist das gewünschte Ergebnis?

- Abhängigkeiten wurden erkannt und keine externen Abhängigkeiten blockieren die Fertigstellung der US?
- Es sind ausreichend Teammitglieder vorhanden und können der US zugewiesen werden, um diese fertig zu stellen?
- Akzeptanzkriterien der Fertigstellung sind klar und testbar?

### Definition of Done

Dieses Dokument legt fest, wann eine User Story als „fertig“ klassifiziert wird und ist eine Art Kriterienkatalog, der angewendet wird.

- Stimmt das Design? (GUI)
- Ist der Code komplett?
  - Hab ich die Codekonventionen eingehalten?
  - Ist der Code verständlich?
  - Ist der Code erweiterbar?
  - Ist der Code kommentiert?
  - Ist der Code im Repository vorhanden?
- Ist der Code getestet?
  - Unittest?
  - Integrationstest?
  - Regressionstest?
  - Systemtest?
  - Sprachtest?
- Wurde eine entsprechende Dokumentation im Testdokument angefertigt?
- Ist die Funktion im Benutzerhandbuch festgehalten?
- Sind alle bekannten Fehler und Mängel behoben (Bugfixing)?
- Akzeptanztest? (durch den Product Owner)

## 2.3. Angewandte Software

Während der Projektarbeit haben wir ausgewählte Software angewandt, die uns bei der Entwicklung unserer Benutzungsoberfläche dienen soll. Dies beinhaltet Entwicklungsumgebungen sowie Tools zur Unterstützung der Organisation im Team und des Projektmanagements. Nachfolgend werden die ausgewählten Software aufgeführt und erläutert.

### 2.3.1. Bitbucket

Bitbucket kommt im Projekt für die Versionskontrolle über Mercurial (siehe 2.3.7) zum Einsatz. Für die Nutzung hat sich das Projektteam aus einer Vielzahl von Gründen entschieden. Durch die Nutzung von Bitbucket entfällt der Bedarf nach einem eigenen Server, den die Projektbetreuung zur Verfügung hätte stellen müssen. Die Nutzung eines eigenen Servers wäre darüber hinaus mit einem Zeit- und Konfigurationsaufwand verbunden. Bei Bitbucket entfallen derartige Aufgaben, sodass die eingesparten Ressourcen von Beginn an in die Entwicklung investiert werden können.

Des Weiteren bietet Bitbucket die Möglichkeit, Teams für ein jeweiliges Repository zu bilden und jedem Teammitglied verschiedene Rechte zu übertragen. Nutzer können hierzu direkt über den Nutzernamen oder über die registrierte E-Mail zum Repository hinzugefügt werden. Ist eine Person noch nicht bei Bitbucket registriert, lässt sich an eine E-Mail-Adresse eine Einladung versenden. Nach dem Hinzufügen zum Repository erhalten die Nutzer alle benötigten Informationen, um beispielsweise das Repository zu klonen. Dies senkt den Organisationsaufwand im Team deutlich.

Bitbucket bietet außerdem gute Übersichten für den Quellcode, die Commits und die Code-Änderungen, sodass Code und Entwicklungsvorgänge zentral über die Webseite betrachtet beziehungsweise nachvollzogen werden können. Dies ist besonders für Open Source-Software von Vorteil, da jeder interessierte Nutzer den Quellcode und die Änderungen betrachten kann. Weitere Vorteile für Open Source-Projekte sind die Möglichkeit für *Pull Requests* sowie das Issue-System. Über die *Pull Requests* kann jeder Interessent Code-Änderungen oder -Erweiterungen melden und um die Aufnahme in den Quellcode bitten. Das Issue-System erlaubt das Melden von Bugs oder Verbesserungen in Form von Tickets, wobei sich Nutzer mittels Kommentarfunktion über das gemeldete Ticket austauschen können.

In einem Wiki können Informationen zur Software sowie zur Nutzung der Software bereitgestellt werden. Auch ein Download-Bereich ist vorhanden, über welchen zum Beispiel die Verteilung aktueller Builds der Software möglich ist. Ein Repository ist zudem an einen beliebigen Bitbucket-Nutzer übertragbar, sodass das Repository nach dem Projektabschluss an die Betreuung abgegeben werden kann.

Wie sich erkennen lässt, bietet Bitbucket weit mehr als nur einfache Versionskontrolle und ermöglicht die volle Konzentration auf die Entwicklung der Software.

### 2.3.2. Confluence

Confluence ist ein auf Java- und webbasiertes Wiki-System oder auch Hypertext-System, das der Projektgruppe zur Kommunikation, zur kollaborativen Textbearbeitung und zum Austausch von Daten, Wissen und Organisationsaspekten dient. Hauptarbeitsoberfläche von Confluence ist das sogenannte Dashboard, in dem der Nutzer über die Auflistungen und Verlinkungen der ihm erlaubten Arbeitsbereiche verfügt. Diese Adressräume können zudem um weitere, eigen kreierte Adressräume niederer Kategorien erweitert werden. Dazu können ebenfalls neue Ebenen angelegt werden. Dies erlaubt eine sinnvolle Kategorisierung von zu kommunizierenden Themen und Aufgabenbereichen.

Weitere Werkzeuge wie die Suche, Änderungsanzeige und Favoritenangabe erweitern diese Funktionalität. Werden einzelne Seiten für sich betrachtet, können neben üblichen Vorgängen wie der Text-, Überschriften- und Formaterstellung auch Listen erstellt, Bilder und externe Dateien eingefügt und Inhalte verwaltet werden. Ein Vorteil von Confluence ist die Einbindung diverser Schnittstellen wie die Makroerstellung, Drag-and-Drop-Funktionalität und weitere. Daraus ergibt sich eine zunehmend vereinfachte Kommunikationsgrundlage.

Daher wurde sich ergänzend zu einem internen Bitbucket-Ordner und in erster Linie unterstützend zur Projektmanagementsoftware *Jira* für Atlassians Confluence entschieden. Confluence dient der Projektgruppe zum Wissensaustausch, da sie mit angewandten Möglichkeiten einen vereinfachten Austausch, eine kollaborative Bearbeitung von Texten und weitere Funktionen ermöglicht. Darüber hinaus ist das angebotene Gesamtpaket Atlassians, indem Jira und Confluence enthalten sind und die durch das OFFIS ermöglichte lizenzkostenfreie Nutzung ein weiteres Entscheidungskriterium.

### 2.3.3. Deployment-Tools

Für das Deployment kommen im Projekt ausgewählte Werkzeuge zum Einsatz, die in der Python-Community weit verbreitet, beziehungsweise standardisiert und von der *Python Packaging Authority*<sup>1</sup> empfohlen sind. Dazu zählen folgende Werkzeuge:

- **pip**: Das Werkzeug *pip* dient für das Paketmanagement. Mit *pip* können Pakete zum Beispiel installiert, deinstalliert, aktualisiert oder aufgelistet werden. Somit lässt sich die korrekte Funktionsweise der eigenen Softwarepakete mittels *pip* testen.

<sup>1</sup><https://packaging.python.org/en/latest/>



- **setuptools:** *Setuptools* stellt ebenfalls ein Werkzeug für die Paketverwaltung dar und wird zudem für die Paketerstellung im Projekt genutzt.
- **wheel:** Mit dem Werkzeug *wheel* können ebenfalls Softwarepakete erstellt werden. Das Werkzeug soll in Zukunft die Paketerstellung mittels *setuptools* vollständig ersetzen.
- **virtualenv:** *Virtualenv* wird zur Erstellung virtueller Python-Umgebungen genutzt, die unabhängig von der System-Installation von Python und anderen Python-Umgebungen sind. Somit sind in jeder virtuellen Umgebung nur die für einen bestimmten Zweck benötigten Pakete installierbar, sodass keine Beeinträchtigung anderer Python-Umgebungen stattfindet. Im Projekt dienen virtuelle Umgebungen dazu, die korrekte Installation und Funktionalität der eigenen Software sowie die Kompatibilität zu verschiedenen Versionen eingesetzter Pakete zu testen.
- **twine:** Das Werkzeug *twine* dient zur Veröffentlichung der eigenen Softwarepakete im *Python Packaging Index*.
- **cx\_Freeze:** Mittels *cx\_Freeze* wird die Software in eine native Applikation für die Betriebssysteme Windows, Linux und Mac OS übersetzt, sodass zur Ausführung keine Installation von Python und benötigten Paketen erforderlich ist.
- **Sphinx:** *Sphinx* dient zur Autogenerierung einer Code-Dokumentation, die über das Web abgerufen werden kann.

#### 2.3.4. Force Atlas 2

Die von der Projektgruppe Maverig umgesetzte Software bietet die Möglichkeit, verschiedenartige Gebäude zu positionieren und diese mit Spannungsleitungen miteinander zu verbinden. Übertragen auf die Mathematik entsteht so ein Graph, der sich aus einer Menge von Knoten und die untereinander verbindenden Kanten zusammensetzt. Die Eingabe solcher Elemente durch den Nutzer ermöglicht zwar die Erstellung eines solchen Graphen, allerdings lassen sich ihre Elemente zusätzlich dermaßen positionieren, dass in der weiteren Arbeit mit solchen Gebilden Vorteile entstehen können. Eine solche Neuordnung wird Graph-Layout genannt. Vorteile sind unter anderem: die Strukturierung der bestehenden, noch nicht geordneten Graphen und die Verbesserung der visuellen Übersicht und des Verständnisses allgemein [Fobo7a]. Als eine der aktuell geeignetsten Methoden, um ein Graph-Layout auf Basis eines unstrukturierten Gebildes zu erstellen, hat sich der Force Atlas-Algorithmus erwiesen: Er ist eine kräftebasierte, auf Knoten arbeitende Strategie. Umschrieben findet die Neuberechnung einer Struktur auf Berechnungen auf



den Knoten statt, indem ihnen zugeordnete Anziehungs- und Abstoßungskräfte mit den anderen Punkten untereinander ihre jeweiligen Abstände berechnen [Kla14a]. Der Algorithmus wird in der aktuellsten Version Nummer zwei verwendet [GEP11]. Als mehrfache Weiterentwicklung von kräftebasierten Knoten-Algorithmen werden eine Vielzahl von Layoutkriterien erfüllt, sodass als Ergebnis eine Anordnung hoher Güte entsteht. Derartige Layoutkriterien können nach [Fob07b] darstellen:

**closeness** Stärkere Anziehung verbundener Knoten

**smallest separation** Mindestabstände von Knoten

**fixed edge length** Einheitliche Kantenlängen

**symmetry** Symmetrie im Gesamtgraphen

**uniform distribution** Gleichmäßige Knotenverteilung im Zeichenfenster

**adaption to the frame** Graph-Format ans Zeichenfenster angepasst

**edge crossing minimization** Kantenkreuzungen minimal halten

**edge directions** Richtungen von Kanten berücksichtigen

Zusätzlich zum Layouting kann dieser Algorithmus durch unterschiedliche Skalierung, Schwerkräfte von Anziehungen, Fenstergrößenmanagement und Clusterverstärkungen differenzierte Varianten erstellen, wenn dies vom Nutzer gewünscht ist. Es ist also Interaktivität und Parametrisierung von Ergebnissen möglich [Kla14b].

### 2.3.5. JIRA

Jira ist eine auf Java- und webbasierte Anwendung zur Fehlerverwaltung, Problembehandlung und in erster Linie eine Projektmanagementsoftware (PM-Software) für die Softwareentwicklung. Aufgrund der Vielzahl am Markt verfügbaren PM-Software erfolgte eine ausgiebige Recherche im Vorfeld der Projektgruppe im Rahmen einer Seminarphase. Daraus resultierte eine kleine Vorauswahl mithilfe von Erfahrungswerten und ein anschließender Vergleich anhand Projektgruppen-relevanter Eigenschaften. Zu diesen Eigenschaften gehören eine webbasierte Anwendung, die Lizenzbestimmungen, Funktionen einer kollaborativen Software, Scheduling, Issue-Tracking (Tickets) und im speziellen die Eignung und Unterstützung von *Scrum* als Vorgehensmodell für das Projektmanagement.

In der Softwareentwicklung unterstützt Jira das Anforderungsmanagement, die Statusverfolgung und den Fehlerbehebungsprozess. Die Hauptarbeitsfläche von Jira ist

das Dashboard, in dem der Nutzer die Auflistungen und Verlinkungen der ihm erlaubten Arbeitsbereiche verfügt. Im Wesentlichen wird Jira durch Tickets, auch Issues genannt, genutzt. Auf der einen Seite können Tickets zu Projekten oder einzelnen Komponenten zugeordnet werden. Auf der anderen Seite können Tickets eine Zusammenfassung, einen Typ, einen Status, eine Priorität, einen Inhalt, Anhänge, Kommentare und weitere Informationen sowie selbst definierte Felder beinhalten. Informationen und Eigenschaften der Tickets lassen sich bearbeiten, der Status eines Tickets kann gewechselt werden und entspricht somit dem aktuellen Bearbeitungsstand.

Des Weiteren unterstützt Jira agile Vorgehensmodelle zur Softwareentwicklung und im speziellen Kanban und Scrum über das entsprechende zur Verfügung stehende Board. Im Scrum Board wird hauptsächlich zwischen den Reitern *Plan*, *Arbeit* und *Bericht* unterschieden. Im ersten Bereich kann das Projekt mithilfe von Sprints und Epen strukturiert und geplant werden. Dazu wird eine komplette Übersicht des Projektes dargestellt. Der zweite Reiter stellt alle noch offenen und bereits fertiggestellten Aufgaben eines Sprints gegenüber. Der letzte Bereich liefert unterschiedliche Informationen zum Projekt in Form von Burndown-Diagrammen, Sprint- und Epos-Berichten sowie weiteren Darstellungsarten.

Abschließend stellt Jira eine große Anzahl von Konfigurationsmöglichkeiten und eine umfangreiche anzupassende Architektur bereit, die einen Einsatz für eine Vielzahl von Zwecken, wie Fehler, Aufgaben, Anforderungen usw. ermöglichen.

Bezug nehmend auf den in der Seminarphase erarbeiteten Vergleich verschiedener PM-Software bietet Jira in diesem Fall einen ähnlichen Funktionsumfang im Vergleich zu untersuchten Alternativen, wie *Redmine*, *OpenProject* und *Trac*. Dazu bezieht die an Redmine angelehnte Entwicklung Jiras mit einer ähnlichen Optik und einer zusätzlich an Scrum angepassten Funktionalität. Da Redmine aufbauend auf Trac entwickelt wurde und Trac aufgrund des Softwareprojekts bei einem Großteil der Projektgruppen-Mitglieder bekannt ist, fiel die Entscheidung auf Jira. Ein weiterer Grund ist das angebotene Gesamtpaket Atlassian, in dem Jira und Confluence enthalten sind und die lizenzkostenfreie Nutzung, die durch das OFFIS ermöglicht wurde.

### 2.3.6. $\LaTeX$

$\LaTeX$  ist ein Textverarbeitungsprogramm und dient zur Darstellung aller Erarbeitungen und Dokumentationen des Projektes, inklusive dieses Abschlussberichtes. Die Nutzung dieser Lösung umfasst eine Distribution (MiKTeX) und eine Entwicklungsumgebung (TeXnicCenter). Im Abgleich mit verwandten Programmen ist der Vorteil von  $\LaTeX$ , dass die Struktur des erzeugten Inhaltes sich in vereinzelt Dateien unterteilen lässt. Dies ermöglicht eine multinutzerorientierte Arbeit, welche erlaubt, dass parallel an verschiedenen Textpassagen gearbeitet werden kann, oh-



ne dass Versionskonflikte entstehen. Technisch betrachtet geschieht dies über sogenannte input-Befehle von .tex-Dateien in eine Hauptdatei, wobei eine solche Unterstrukturierung auch feingranularer stattfinden kann. Desweiteren sind include-Befehle innerhalb von .tex-Dateien nützlich, externe Dateien wie Bilder und PDF-Dokumente einzubinden. So müssen Externdateien nicht im Dokument an sich verändert werden, falls Bedarf besteht, sondern direkt in der Datei, welche schlussendlich beim Kompilieren eingebunden wird. Dateiorientierte Arbeit mit Texten, wie in  $\LaTeX$ , ermöglicht als weiteren wichtigen Punkt, dass sich vereinfacht Grundgerüste wie Dokumentvorlagen erstellen lassen, die universell wiederverwendbar sind. Diese Gründe haben die Projektgruppe dazu bewogen, sich für  $\LaTeX$  als Textverarbeitungsprogramm, zu entscheiden.

### 2.3.7. Mercurial

Die Benutzung von Versionskontrollsystemen in der Softwareentwicklung ist bei kleinen sowie großen Unterfangen unumgänglich, da mit ihrer Hilfe wesentliche Aufgaben unterstützt und teilweise erst ermöglicht werden. Zu diesen Aufgaben zählen beispielsweise die Protokollierung der Dateihistorien, Mehrbenutzerverwaltung, Wiederherstellung früherer Revisionen, das Auflösen von Konfliktfällen und die allgemeine Archivierung. Es existiert eine große Menge an unterschiedlichen Versionskontrollsystemen, deren Systemarchitekturen und Nutzungen teilweise stark voneinander abweichen. Allgemein lässt sich sagen, dass ein Wissen über die Funktionsweise und der sichere Umgang mit den gängigen Begrifflichkeiten unabdingbar für das Entwickeln mit Hilfe dieser Systeme ist. Das Versionskontrollsystem Mercurial stellt eine moderne Variante mit verteilter Architektur sowie einfacher Lernkurve dar und bietet sich daher für die Nutzung in der Projektgruppe Maverig an. Insbesondere für Entwickler, die nicht bereits jahrelang den Umgang mit Versionskontrollsystemen auf Basis einer Client-Server-Architektur gewohnt sind, ist das Erlernen und die Nutzung eines verteilten Versionskontrollsystems sinnvoll.

### 2.3.8. Mosaik

Mosaik<sup>2</sup> ist ein modulares Simulationsframework zur Komposition und Simulation von Smart Grid Szenarien, welches am OFFIS entwickelt wurde. Mosaik bietet eine Schnittstelle für die Einbindung von virtuellen Energiekomponenten und Kontrollstrategien mithilfe von sogenannte Simulatoren an. Ein Simulator kapselt die technische Implementierung energiespezifischer oder logischer Modelle.

Die Schnittstelle kann dabei über alle Programmiersprachen angesprochen werden, die das Datenaustauschformat JSON unterstützen. Der Simulations-Manager

---

<sup>2</sup>Mosaik Website: <http://mosaik.offis.de>



von Mosaik kann dabei auch bereits laufende oder auf anderen Servern liegende Simulator-Prozesse einbinden. Außerdem gibt es für Python eine *High-Level-Api* von der für die Implementierung von Simulatoren direkt abgeleitet werden kann.

Es gibt bereits einige Simulator-Implementierungen, welche im Folgenden aufgelistet sind:

- **PyPower-Simulator**: unterstützt die grundlegenden Stromnetzkomponenten (Referenzknoten, Transformatoren, Netzknoten, Kabel und Leitungen) und simuliert den Energiefluss. Die Topologie der Stromnetzkomponenten kann im Excel- oder JSON-Dateiformat übergeben werden.
- **Household-Simulator**: simuliert den Energieverbrauch von Haushalten anhand vorgegebener Datenreihen.
- **CSV-Simulator**: simuliert den Energieverbrauch einer Energiekomponente (z.B. Photovoltaic) anhand vorgegebener Datenreihen.
- **MAS-Simulator**: unterstützt ein Dummy-Multi-Agentensystem zur Veranschaulichung der Schnittstellenkapazitäten im Bezug auf Multi-Agenten-Systeme.
- **WebVis-Simulator**: enthält ein Modell, welches während der Simulation Daten anderer Entitäten (Modellinstanzen) abfragt und diese mithilfe eines Webinterfaces visualisiert.
- **HDF5-Simulator**: enthält ein Modell, welches während der Simulation Daten anderer Entitäten protokolliert und diese in einer HDF5-Datenbank speichert.

Um die Simulation zu starten, müssen die Simulatoren und deren Entitäten initialisiert und gestartet werden. Anschließend können die Entitäten in Mosaik miteinander verbunden werden.

Nach der Szenario-Komposition, kann nun die Simulation in Mosaik ausgeführt werden. Während der Simulationslaufzeit kann eine Entität auch auf Daten anderer Entitäten zugreifen, zu denen eine Verbindung besteht. Beispiele hierfür sind die oben genannten Simulatoren MAS, WebVis und HDF5. Diese sammeln und verarbeiten beispielsweise Daten verschiedener Energiekomponenten der Stromnetztopologie.

Für die oben genannten Simulator-Implementierungen (ohne MAS) gibt es ein **Demo-Szenario**<sup>3</sup>, welches die Komposition und Simulation mittels Mosaik veranschaulicht.

Mosaik ist quelloffen unter der LGPL-Lizenz verfügbar und kann mithilfe des Python Package Index (pip) **installiert**<sup>4</sup> werden.

<sup>3</sup>Demo-Szenario: <https://bitbucket.org/mosaik/mosaik-demo/>

<sup>4</sup>Mosaik Installation: <https://mosaik.offis.de/install>

### 2.3.9. Poedit

Poedit ist ein kostenloses Übersetzungsprogramm für Software, die die Funktion gettext unterstützen. Mit Poedit ist es möglich bestimmte Wörter/ Sätze in andere Sprachen zu übersetzen um die Software zu internationalisieren. Das Programm durchläuft automatisch den ganzen Sourcecode und filtert nach vordefinierten Platzhaltern und liefert die Ergebnisse übersichtlich in einer Liste. Die dort aufgelisteten Wörter/ Sätze können nun in beliebige Sprachen übersetzt werden. Ist die Übersetzung abgeschlossen, dann wird nach dem Speichern eine Po-Datei und eine Mo-Datei generiert. Die Po-Datei dient zur späteren Bearbeitung der Übersetzungen, die Mo-Datei hingegen ist für die Funktion gettext, die im Sourcecode die Wörter/ Sätze für die jeweilige Sprache austauscht. Poedit ist besonders für größere Projekte sehr gut geeignet, da die zu übersetzenden Wörter/ Sätze automatisch erkannt und übersichtlich dargestellt werden.

### 2.3.10. PyCharm

Integrierte Entwicklungsumgebungen (IDE) sind hilfreiche Werkzeuge, die den Softwareentwicklern häufig wiederkehrende Arbeiten abnehmen und wichtige benötigte Funktionen schnell und in einer Benutzeroberfläche bereitstellen. Die Idee ist, dass der Entwickler von formalen Arbeiten entlastet wird und sich ganz auf seine Entwicklertätigkeit konzentrieren kann. Für ein Softwareprojekt macht es Sinn sich auf eine IDE festzulegen. Die Projektgruppe hat sich daher aufgrund der Vorteile auf die IDE PyCharm festgelegt. PyCharm, eine IDE für Python, wurde von der Firma JetBrains entwickelt. Mit IntelliJ IDEA bietet die Firma JetBrains auch eine bekannte und leistungsstarke Alternative für die Programmiersprache Java an. Für die Programmierung in Python unterstützt PyCharm den Softwareentwickler mit hilfreichen Funktionen (z.B. Auto-Completion oder Refactoring) und ist somit eine gute Entscheidung für die Projektgruppe.

### 2.3.11. PySide

PySide ist ein Python-Binding für das plattformübergreifende Toolkit Qt, das zur GUI-Programmierung eingesetzt wird. PySide unterstützt die Qt-Version 4, im Gegensatz zu PyQt (alternatives Binding für Qt) ist PySide allerdings unter der LGPL veröffentlicht [QT12]. Da PySide einen größeren Funktionsumfang als die Standardbibliothek von Python bietet, ist es für größere Projekte geeigneter. Daher und aus lizenzrechtlichen Gründen hat sich die Projektgruppe Maverig dazu entschieden, PySide für die Entwicklung der grafischen Benutzeroberfläche zu benutzen.



## 2.4. Softwarelizenzierung

Bevor eine Software auf den Markt gebracht werden kann, muss entschieden werden, unter welcher Lizenz diese vertrieben werden soll. Nachfolgend werden wir näher darauf eingehen, welche mögliche Lizenzen zur Wahl stehen und unter welcher Lizenz unsere Software am Ende vertrieben wird.

### 2.4.1. Vorgehensweise für die Softwarelizenzierung

Wenn Entwickler eine Software auf den Markt bringen möchten, so stehen sie unter anderem vor der Wahl der Lizenz für ihr Produkt. Dabei muss der Entwickler grundsätzlich die Entscheidung treffen, ob seine Software als proprietäre (Closed-Source) Software vertrieben werden soll und somit der Quellcode anderen Entwicklern zur Modifikation und Weiterentwicklung nicht mehr zur Verfügung steht oder ob sie als Open-Source Software für jeden nutzbar ist. Open-Source hat den Vorteil, dass die Software von einer Vielzahl an Entwicklern modifiziert und erweitert werden kann, sodass die Weiterentwicklung und Verbesserung des Produktes wahrscheinlich ist. Zur Vermeidung der Überführung einer Open-Source Software von anderen Entwicklern in die proprietäre Domäne, muss der Entwickler die richtige Wahl der Software-Lizensierung treffen. Eine Softwarelizenz sagt aus, welche Rechte der Anwender bei der Nutzung der Software besitzt. Manifestiert ist dies im sogenannten Lizenzvertrag. Eine der bekanntesten Softwarelizenzen ist die GNU General Public License. Sofern eine Software unter dieser Lizenz steht, muss sie auch nach Modifikation und Erweiterung unter dieser verwendet werden [GNU14a]. Weniger streng sind dagegen die Lizenzen LGPL oder BSD, welche den Anwendern erlauben, die Software unter anderem in proprietäre Software einzubinden. Vor der Wahl einer passenden Lizenz für seine Software muss sich der Entwickler informieren, unter welchen Lizenzen die von ihm verwendeten Frameworks und Libraries stehen, damit er keine Lizenzvereinbarungen verletzt. Zudem sollte der Entwickler vorher entscheiden, ob seine Software Open-Source oder Closed-Source Software sein soll und welche Freiheiten er zukünftigen Anwendern an seiner Software zugestehen möchte.

### 2.4.2. Wahl der Software-Lizenz

In der Projektgruppe *Maverig* bestand freie Wahl beim Lizenzierungsmodell, da die verwendete Software von den eingebundenen Libraries und Frameworks wie Mosaik (LGPL), Qt Pyside (LGPL), numpy (BSD), NetworkX (BSD) und python-dateutil (BSD) dem schwachen Copyleft unterliegt und somit keine großen Lizenz einschränkungen vorlagen. Aus diesem Grund fiel die Entscheidung auf die Lesser General Public License (LGPL), welche eine abgeschwächte Variante der General Public Li-



cense (GPL) ist. Im Gegensatz zur GPL darf die Software, die der LGPL unterliegt, in proprietäre Programme eingebunden werden. Die LGPL bietet generell mehr Freiheitsrechte und weniger Einschränkungen für den Anwender [GNU14b]. Dies kann zu einer erhöhten Nutzung und Förderung der Popularität der Software führen. Aufgrund der geringen Einschränkungen wird das Erreichen einer lebendigen Community erhofft, die gemeinsam die Visualisierung von Smart Grid Simulationen stark vorantreibt, sodass eine umfangreiche, robuste sowie qualitativ hochwertige Software entsteht.

## 3. Grundlagen

Da die Projektgruppe Maverig im Kontext von Smart Grids (intelligentes Netzmanagement) und erneuerbaren Energien arbeitet, ist es wichtig ein Grundverständnis über die Arbeit von Energiesystemen und die Simulation von Smart Grids zu haben. Das Grundverständnis reicht von technischen Aspekten wie die Erzeugung elektrischer Energie, deren Transport und Übertragung hin zu dem Systemmodell von Smart Grids und den Herausforderungen im aktuellen Energiesektor. Diese Grundlagen sollen in diesem Kapitel vermittelt werden.

### 3.1. Arbeit von Energiesystemen

Um die grundlegende Arbeit von Energiesystemen zu verstehen, werden in diesem Abschnitt zuerst wichtige physikalische Grundbegriffe und die Erzeugung elektrischer Energie erklärt. Weiterhin wird die Übertragung, der Transport und die Verteilung elektrischer Energie behandelt, bevor dann verschiedene Netztopologien und von den Netzbetreibern zu erbringende Systemdienstleistungen vorgestellt werden.

#### 3.1.1. Physikalische Grundbegriffe

Im Folgenden werden die wichtigsten physikalischen Grundbegriffe erläutert.

##### Elektrischer Strom

Der elektrische Strom ist die gezielte und gerichtete Bewegung freier Ladungsträger. Allgemein wird die elektrische Stromstärke mit dem Formelzeichen  $I$  gekennzeichnet und in der Einheit A (Ampere) gemessen. Seltener wird sie auch als Stromstärke oder Strommenge bezeichnet [ELK15a].

##### Elektrische Spannung

Elektrische Spannung ist eine physikalische Grundgröße, die angibt, wie viel Energie nötig ist um eine elektrische Ladung innerhalb eines elektrischen Feldes zu bewegen. Das Formelzeichen der Spannung ist  $U$ . Spannung wird in der Einheit V (Volt) gemessen [ELK15b].



### Elektrische Leistung

Elektrische Leistung als physikalische Größe bezeichnet die in einer Zeitspanne umgesetzte elektrische Energie bezogen auf diese Zeitspanne. Formelzeichen der elektrischen Leistung ist das  $P$ , sie wird in der Einheit  $W$  (Watt) gemessen [ELK15c].

### Elektrische Energie

Elektrische Energie bezeichnet Energie, die mittels Elektrizität übertragen wird, man spricht hier von elektrischer Arbeit. In der Energiewirtschaft ist die Maßeinheit kWh (Kilowattstunde) üblich. Sie setzt sich aus der Spannung ( $U$ ), dem Strom ( $I$ ) und der Zeit ( $t$ ) zusammen [ELK15d].

### Drehstrom

Die elektrische Energie wird in Elektroenergiesystemen als sog. Drehstrom übertragen. Drehstrom besteht aus 3 einzelnen Strömen gleicher Frequenz, die zueinander phasenverschoben sind. Aufgrund dieser Phasenverschiebung ist der Einsatz eines Neutralleiters nicht nötig, der Materialeinsatz halbiert sich (pro Phase Einsparung eines Neutralleiters) [ELK15e].

### Netzfrequenz

Die Netzfrequenz gibt den Polaritätswechsel der Spannung pro Sekunde an und wird in Hertz gemessen. Sie beträgt in europäischen Netzen im Optimalfall genau 50 Hertz. Diese 50 Hertz können aber nur garantiert werden, wenn die Erzeugerleistung mit der Verbraucherleistung übereinstimmt. Bei zu geringer Erzeuger- bzw. zu hoher Verbraucherleistung sinkt die Frequenz et vice versa. Da sich elektrische Energie nur bedingt zwischenspeichern lässt, muss zu der aktuellen Verbraucherleistung eine gleich große Erzeugerleistung gegenüberstehen um die Frequenz konstant zu halten. Dies ist unter anderem eine Aufgabe der Systemdienstleister.

### 3.1.2. Erzeugung elektrischer Energie

Als Erzeugung von elektrischer Energie versteht man die Umwandlung eines Primärenergieträgers wie z.B. Kohle, Wasser, Sonne oder Wind in elektrische Energie. In diesem Kapitel soll auf die verschiedenen Arten der Erzeugung von elektrischer Energie eingegangen werden. Es wird hier unterscheiden zwischen Atomkraftwerken, Wärmekraftwerken, Wasserkraftwerken und erneuerbaren Energien.

### Wärme kraftwerke

Zu den Wärme kraftwerken zählen die Dampf-, Gasturbinen- sowie Dieselm aschinenkraftwerke. In diesen Kraftwerken wird thermische Energie (Wärme) in elektrische Energie umgewandelt. Diese Umwandlung erfolgt meist über den Zwischenschritt der Erzeugung von kinetischer Energie durch eine Wärme kraftmaschine. Anschließend erfolgt die Umwandlung in elektrische Energie durch den Betrieb eines Generators. Wärme kraftwerke besitzen die Eigenschaft der deterministischen Verfügbarkeit, sie sind also rund um die Uhr mit einer festen Leistung einplanbar, im Gegensatz zu den meisten erneuerbaren Energien. Sie eignen sich daher ideal für die Grundlast eines Energienetzes.

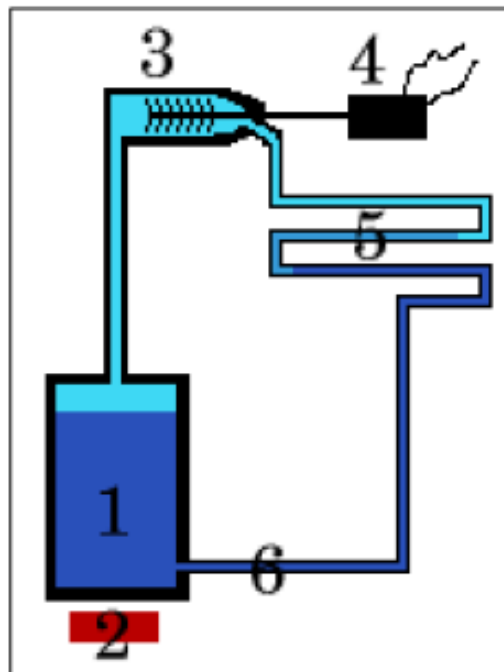


Abbildung 3.1.: Funktionsprinzip eines Wärme kraftwerks [Baro4]

Abbildung 3.1 zeigt eine schematische Darstellung des Funktionsprinzips eines Wärme kraftwerks. Im Dampfkessel (1) befindet sich Wasser. Durch das Hinzufügen von thermischer Energie (2) beginnt dieses Wasser zu verdampfen. Der Wasserdampf wird hierbei durch eine Turbine (3) geleitet. Diese wiederum treibt einen Generator (4) an, der aus der kinetischen Energie der Turbine elektrische Energie erzeugt. Anschließend wird der abgekühlte Wasserdampf durch einen Kondensator (5) soweit heruntergekühlt, dass er im flüssigen Aggregatzustand dem Dampfkessel erneut zugeführt werden kann (6) [Sch12a].

### Kernkraftwerke

Grundsätzlich zählen die Kernkraftwerke ebenfalls zu den Wärmekraftwerken. Die Wärmeerzeugung erfolgt allerdings nicht durch die Verbrennung von Rohstoffen sondern durch das Freisetzen von Kernenergie. Kernkraftwerke bestehen im wesentlichen aus einem Reaktorkern. In diesem erfolgt die Kernspaltung und die Freisetzung von Energie. Anschließend wird durch einen nachgeschalteten Wasserdampfkreislauf eine Turbine angetrieben, welche wiederum einen Generator antreibt um elektrische Energie zu erzeugen.

### Wasserkraftwerke

Wasserkraftwerke nutzen als Primärenergieträger Wasser, sind also eigentlich den erneuerbaren Energien zuzuordnen. Aufgrund ihrer deterministischen Verfügbarkeit werden sie aber als autonome Kraftwerksklasse aufgeführt. Laufwasser-, Speicher-, Pumpspeicher- und Gezeitenkraftwerke zählen zu der Klasse der Wasserkraftwerke. Wasserkraftwerke wandeln die potentielle Energie des Wasser, durch den Einsatz einer Turbine bzw. Wasserrades und die Nutzung eines Generators, in elektrische Energie um.

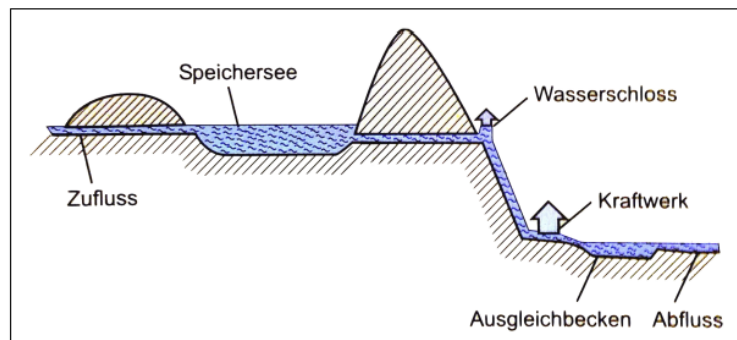


Abbildung 3.2.: Funktionsprinzip eines Wasserkraftwerks [Sch12j]

Prinzipiell besitzen alle Wasserkraftwerke, mit Ausnahme der Gezeitenkraftwerke (Nutzen der kinetischen Energie der Wasserströmung, verursacht durch die Gezeiten), eine Staumauer um Wasser auf einem höheren Niveau zurückzuhalten. Dessen Funktionsprinzip ist in Abbildung 3.2 dargestellt. Durch das Abfließen des Wassers über eine Turbine erfolgt die Umwandlung in kinetische Energie, gefolgt von der Umwandlung in elektrische Energie durch einen Generator. Speicherkraftwerke und Pumpkraftwerke besitzen zusätzlich einen Speichersee, in dem das gestaute Wasser gesammelt werden kann. Dies ermöglicht die gezielte und termingenaue Erzeugung von elektrischer Energie z.B. zur Deckung einer Spitzenlast. Pumpkraftwerke ermöglichen es zudem, zu viel vorhandene elektrische Energie in kinetische

Energie umzuwandeln. Pumpen führen dazu Wasser in einen Speichersee auf ein höheres Niveau zurück. Bei Bedarf kann dieses zurückgeführte Wasser wieder zur Erzeugung elektrischer Energie benutzt werden.

### Erneuerbare Energien

Im Gegensatz zu den klassischen Energieerzeugern wie z.B. den Wärmekraftwerken setzen die erneuerbaren Energien nicht auf endliche Primärenergieträger (bspw. fossile Brennstoffe), sondern auf Energieträger, die dem Menschen quasi in unendlicher Menge zur Verfügung stehen wie bspw. (schnell) nachwachsende Rohstoffe. Zu nennen wären hier zudem noch Wind und Sonne. Streng physikalisch gesehen handelt es sich hierbei aber nicht um Primärenergieträger. Windkraftanlagen, Solaranlagen, Biomassekraftwerke, Geothermiekraftwerke und Brennstoffzellen zählen zu den erneuerbaren Energien.

### Windkraftanlagen

Windkraftanlagen wandeln mittels Windturbinen mechanische Rotationsenergie in elektrische Energie um. Früher wurden Windräder vermehrt im Inselbetrieb für eine autarke Energieversorgung genutzt. Heute werden Windräder in Windparks zusammengefasst und speisen elektrische Energie direkt in das Niederspannungsnetz ein, bei größeren Anlagen direkt in das Mittelspannungsnetz. Man unterscheidet zwischen Off- und Onshore-Windparks. Offshore-Windparks befinden sich auf der offenen See vor der Küste, Onshore-Windparks auf dem Festland.

### Solarenergieanlagen

Solarzellen sind in der Lage Solarstrahlung (Sonnenlicht) in elektrische Energie auf Basis des photovoltaischen Effektes umzuwandeln. Mehrere untereinander verschaltete Solarzellen bilden Solarmodule, viele Module eine Photovoltaikanlage. Solarzellen bestehen aus 2 verschiedenen Schichten, einer n- und einer p-Schicht. Aufgrund von Elektronenüberschuss bzw. -mangel tritt eine Potentialdifferenz bzw. Spannung auf. Auf Basis dieser entstehenden Potentialdifferenz kann elektrische Energie erzeugt werden.

### Biomassekraftwerke

Biomassekraftwerke nutzen die als Primärenergieträger nachwachsbaren Rohstoffe aus der Natur. Durch Hitze, entstehend durch die Verfeuerung der Rohstoffe, wird Wasserdampf erzeugt um eine Turbine anzutreiben. Die kinetische Energie wird anschließend mittels eines Generators in elektrische Energie umgewandelt. Aufgrund

der hohen Abwärme bei der Verfeuerung der Rohstoffe werden Biomassekraftwerke häufig mit einem Blockheizkraftwerk kombiniert, wodurch die Abwärme für das Nah- und Fernwärmenetz genutzt wird. Biomassekraftwerke besitzen deterministischen Charakter, lassen sich also ideal für die Grundlastzeugung nutzen, da konstant Energie erzeugt wird, solange Primärenergieträger in Form von Rohstoffen zur Verfügung stehen. Dies unterscheidet die Biomassekraftwerke von Windkraftanlagen und Solarenergieanlagen.

### Geotherme Kraftwerke

Geothermiekraftwerke nutzen die annähernd unbegrenzt verfügbare Erdwärme zur Erzeugung elektrischer Energie. Durch die konstante Verfügbarkeit von Erdwärme sind Geothermiekraftwerke ebenfalls deterministisch und können so für die Grundlast- oder auch Spitzenlastzeugung genutzt werden. Man unterscheidet zwischen Erdwärmesonden, hydrothermalen Systemen und petrothermalen Systemen [Sch12b].

### 3.1.3. Übertragung, Transport und Verteilung von elektrischer Energie

Die Übertragung und der Transport von elektrischer Energie erfolgt über verschiedene Arten von Netzen. Um die Verluste bei der Übertragung von hohen Leistungen möglichst gering zu halten, arbeiten die Netze mit unterschiedlich hohen Spannungen. Je höher die Spannung, desto mehr elektrische Energie kann über eine größere Entfernung (vergleichsweise) verlustfrei transportiert werden. Um Leitungsverluste zu kompensieren wäre theoretisch auch eine Erhöhung des Stromes möglich. Hierzu müsste allerdings der Leitungsquerschnitt stark erhöht werden, was aus wirtschaftlicher Sicht, aufgrund des erhöhten Materialeinsatzes, nicht sinnvoll ist. Die Umwandlung in die verschiedenen Spannungsebenen der Netze erfolgt durch Transformatoren. Diese erhöhen oder verringern das Spannungsniveau je nach benötigter Höhe und sind zwischen den Netzen positioniert. Abbildung 3.3 zeigt den Aufbau eines Energienetzes mit den dazugehörigen Spannungsebenen, die im Folgenden detaillierter erklärt werden.

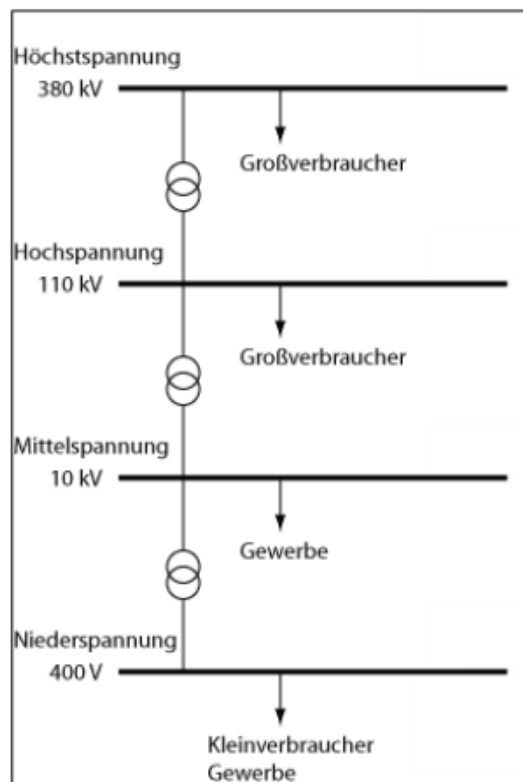


Abbildung 3.3.: Aufbau eines Energienetzes

### Transportnetze

Transportnetze werden als Höchstspannungsnetze (HöS-Netze) bezeichnet und übertragen elektrische Energie mit einer Spannung von 220kV, 380kV und bei sehr großen überregionalen Entfernungen mit einer Spannung von 765kV. In diese Netze speisen die großen überregionalen Kraftwerke ca. 70% der benötigten elektrischen Energie für den öffentlichen Bedarf ein. Das Transportnetz ist als maschenförmiges Netz aufgebaut und ist zusätzlich über Kuppelleitungen an andere (internationale) Netze angeschlossen für den Ausgleich von Energieüberschuss bzw. -mangel. Die Energieübertragung erfolgt in wechselnder Richtung [Sch12c].

### Übertragungsnetze

Übertragungsnetze, auch Hochspannungsnetze (HoS-Netze), arbeiten mit einer Spannung von 110kV und übernehmen die Verteilung von Energie ab dem Entnahmepunkt aus dem Transportnetz zu kleineren Verteilerunternehmen oder Großabnehmern. Sie sind als Maschen- oder Strahlennetz aufgebaut. Im Gegensatz zu den Transportnetzen können sie Transportfunktion oder Verteilerfunktion einnehmen.

Im ersten Fall ist die Energieflussrichtung unbestimmt. Bei der Verteilerfunktion ist die Energieflussrichtung hingegen eindeutig zum Verbraucher gerichtet [Sch12d].

### Mittelspannungsnetze

Mittelspannungsnetze sind Verteilungsnetze und arbeiten mit einer Spannung von 10kV/20kV. Sie beziehen ihre elektrische Energie aus den vorgelagerten Übertragungsnetzen und verteilen diese zu Ortsnetzstationen oder Großabnehmern. Diese direkt an das Mittelspannungsnetz angeschlossenen Abnehmer betreiben einen eigenen Umspannungstransformator. Bei der Einspeisung von elektrischer Energie in das Mittelspannungsnetz spricht man von primärer Verteilung, bei der Abgabe von Energie in das Niederspannungsnetz von sekundärer Verteilung [Sch12e].

### Niederspannungsnetze

Das Niederspannungsnetz oder Ortsnetz arbeitet mit einer Spannung von 400V und ist als Strahlen-, Ring- oder Maschennetz aufgebaut. Niederspannungsnetze werden aus dem Mittelspannungsnetz gespeist und sind Verteilernetze. Im Unterschied zu den anderen vorgelagerten Netzen sind Niederspannungsnetze als 4-Leiter-Systeme aufgebaut um eine Nutzung von einphasigen Verbrauchern zu ermöglichen. Bei der Nutzung von nur einer Phase steht die uns bekannte Spannung von 230V zur Verfügung [Sch12f][Sch12g].

#### 3.1.4. Netztopologien

Bei den Netztopologien unterscheidet man zwischen Strahlen-, Ring- und Maschennetz, zusätzlich gibt es noch Mischformen der verschiedenen Typen. Außerdem besitzen diese Netze Trennstellen um bei Überlast, Kurzschluss oder Wartung um spannungsfrei geschaltet und vom Netz getrennt zu werden. Durch diese Trennung kann sich die Netztopologie ändern.

#### Strahlennetz

Im Strahlennetz erfolgt eine zentrale Einspeisung über Sticleitungen. Abgehende Leitungen (Strahlen) versorgen bspw. Verbraucher in Niederspannungsnetzen. Der Vorteil von Strahlungsnetzen liegt in ihrer geringen Komplexität und der daraus folgenden Übersichtlichkeit und des geringen Planungsaufwandes. Nachteilig hingegen ist der komplette Versorgungsverlust der an den Strahlen angeschlossenen Verbraucher bei Ausfall der Sticleitung [Leh13a].

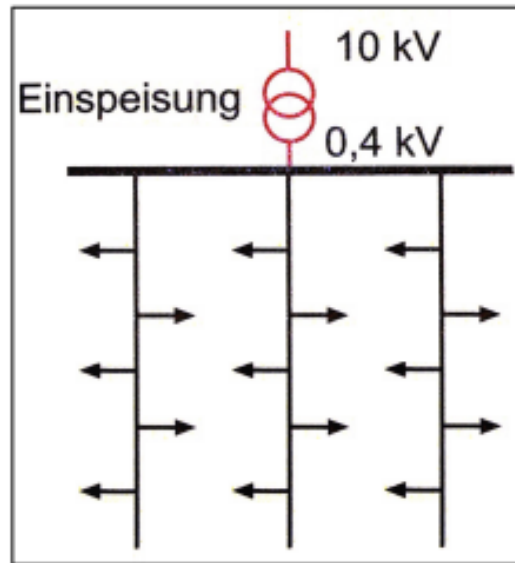


Abbildung 3.4.: Aufbau eines Strahlennetzes [Sch12k]

### Ringnetz

Die Versorgung in einem Ringnetz erfolgt von min. 2 Stellen aus. Die Verbraucher sind über eine Ringleitung bzw. 2 Halbringe an das Netz angeschlossen. Im Störfall kann die Trennstelle den Ring auftrennen. Verbraucher vor dem ungestörten Teilstück können so weiter versorgt werden. Die Vorteile des Ringnetzes liegen in der höheren Versorgungssicherheit sowie verbesserter Spannungshaltung. Nachteilig ist die komplexere Übersicht des Netzes [Sch12h].



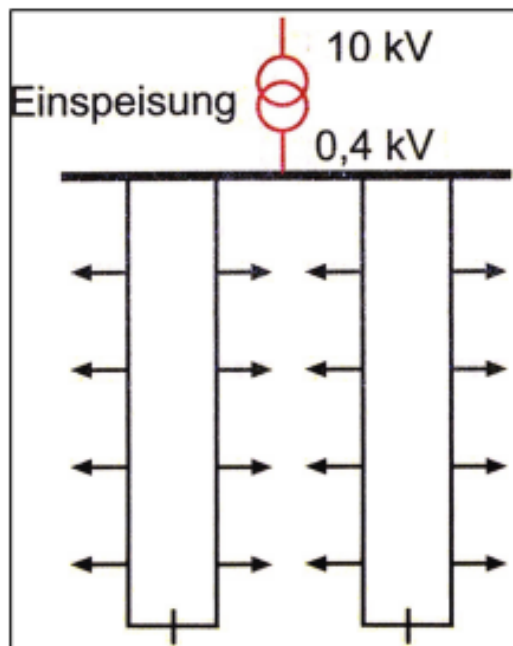


Abbildung 3.5.: Aufbau eines Ringnetzes [Sch12]

### Maschennetz

Maschennetze sind die komplexeste Art der Netztopologien und bestehen aus Knoten und Zweigen die mehrfach versorgt werden. Sie steigern die Versorgungssicherheit im Vergleich zu Ringnetzen erneut. Zusätzlich sind für den Kurzschlussfall alle Knotenpunkte oder Abgänge durch Sicherungen geschützt, wodurch die Auswirkung auf einen kleinen Teil des Netzes begrenzt bleibt. Durch die erhöhte Komplexität und Absicherung verringert sich allerdings die Planbarkeit und Übersichtlichkeit der Maschennetze [Sch12i][Leh13b].

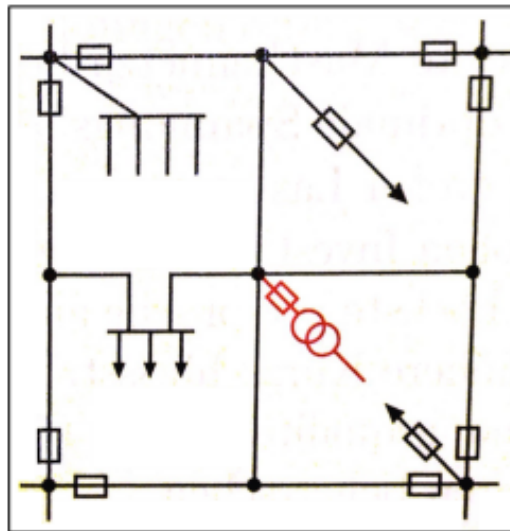


Abbildung 3.6.: Aufbau eines Maschennetzes [Sch12m]

### 3.1.5. Systemdienstleistungen

Damit die elektrische Energie in der vorgegeben Spannung und Frequenz beim Endverbraucher ankommt, sind eine Reihe von Maßnahmen notwendig. Man spricht hier von sog. Systemdienstleistungen. Diese werden von den Transportnetzbetreibern erbracht.

#### Frequenzhaltung

Wie bereits im Abschnitt 3.1.1 zur Netzfrequenz erläutert, ist eine konstante Frequenz nur garantiert bei einem Ausgleich von Energieverbrauch und Energieerzeugung. Die Erzeugung muss den kompletten Energiebedarf abdecken. Dieser ist jedoch nicht konstant und ändert sich über den Tag ständig. So ist der Energiebedarf früh morgens bspw. anders als am Abend. Dieser Wechsel ist jedoch relativ gut vorhersehbar, sodass es Kraftwerke für die Grundlast, Mittellast und die Spitzenlast gibt. Bei Fehlprognosen müssen die Netzbetreiber Regelleistung vorhalten, die bei nicht geplantem Lastanstieg Energie in das Netz einspeisen. Früher erfolgte dies durch Zuschalten von Gas- oder Kohlekraftwerken. Heute versucht man allerdings immer mehr erneuerbare Energien in diesen Prozess einzubinden, so bieten z.B. Pumpspeicherkraftwerke die Möglichkeit Energie aufzunehmen und abzugeben.

### Spannungshaltung

Beim Transport über große Distanzen und dem Verbrauch von elektrischer Energie kommt es zu einem Spannungsabfall. Damit der Spannungsabfall innerhalb der Toleranzen liegt, wird Nahe der Region sog. Blindleistung in das Netz eingespeist. Diese Blindleistung, die nichts zur Wirkleistung beiträgt, wird von Kraftwerken in die Höchst- und Hochspannungsnetze eingespeist. Durch die Energieeinspeisung der erneuerbaren Energien im Nieder- und Mittelspannungsnetz wird zusätzlich die Spannung in diesen Netzen stabilisiert, aufgrund ihrer Nähe zum Endverbraucher.

### Netzengpassmanagement

Bei Energieüberschuss in einer Region wird über die Transportnetze die überschüssige Energie in Regionen mit zu hohem Bedarf geleitet. Hierbei kann es zu Engpässen und Überlastungen des Transportnetzes kommen, wenn bspw. große Mengen Windenergie aus dem Norden in den Süden transportiert werden soll. Das Netzengpassmanagement soll hierfür Maßnahmen zur Verfügung stellen um eine Überlastung zu vermeiden. So können Windparks gedrosselt werden um die Energieerzeugung herunterzufahren. Die überschüssige Energie könnte auch durch Pumpkraftwerke gespeichert werden und zu einer Zeit der Spitzenlast wieder abgerufen werden.

### Versorgungswiederaufbau

Bei einem Netzzusammenbruch durch das Einbrechen der Netzfrequenz wird das Netz in kleine Teile aufgeteilt. Um die Versorgung wiederherzustellen muss das Netz Stück für Stück wieder hochgefahren werden. Ein abruptes Hochfahren sämtlicher Kraftwerke hätte einen erneuten Zusammenbruch aufgrund von Überlastung zur Folge. Zusätzlich benötigen einige Energieerzeuger zum Starten externe Energie. Grundsätzlich erfolgt das Hochfahren des Netzes von der Höchstspannungsebene aus. Durch den Einsatz von erneuerbaren Energie, die sich im Nieder- und Mittelspannungsnetzen befinden, ist aber auch ein Versorgungsaufbau von den unteren Ebenen aus möglich.

## 3.2. Simulation von Smart Grids

Smart Grids lassen Verbraucher und Erzeuger *intelligent* miteinander kommunizieren, um Energienetze zu steuern und zu regeln, sodass Netzkapazitäten ausgenutzt werden können und Lasten optimal verteilt werden. In diesem Abschnitt wird das Thema *Smart Grids* ausführlicher behandelt. Neben dem Vergleich zum konventio-

nellem Netz wird auf die Systemarchitektur, die Kommunikation, auf datenschutzrechtliche Aspekte und aktuelle Herausforderungen eingegangen.

### 3.2.1. Begriffliche Abgrenzungen

Im Energiesektor sind viele neue Begriffe wie zum Beispiel Smart Grids, Smart Homes, Smart Meter entstanden. In diesem Abschnitt sollen diese Begriffe voneinander und insbesondere von der konventionellen Stromversorgung abgegrenzt werden.

#### Konventionelle Stromversorgung

Bevor es darum geht, was *Smart Grids* sind, soll zunächst geklärt werden, wie die Stromversorgung am Beispiel von Deutschland üblicherweise abläuft. So wird in einer Reihe von stromerzeugenden Anlagen, wie beispielsweise Kohlekraftwerken, Wasserkraftwerken, Atomkraftwerken, Wind- und Solaranlagen, Energie erzeugt und mittels Transformatoren und Umspannungsstationen in das Stromnetz eingeführt und verteilt. Diese unterscheiden sich in Hoch-, Mittel- und Niederspannungsnetze. Aber nicht nur kommerzielle Stromerzeuger tragen heutzutage dazu bei Strom in das Netz einzuspeisen. Auch Privatpersonen können mithilfe verschiedener Techniken ihren überschüssig erzeugten Strom speichern oder gegen eine gewisse Vergütung in das Stromnetz einführen. Die verschiedenen Spannungsebenen des Stromnetzes erfüllen auch unterschiedliche Aufgaben. So dient das Hochspannungsnetz zur überregionalen Stromversorgung und der Versorgung von großen Industriebetrieben sowie des Zugverkehrs [Wago3]. Das Mittelspannungsnetz versorgt größere Betriebe und örtliche Verteilerstationen, die wiederum über das Niederspannungsnetz die einzelnen Haushalte versorgen.

#### Smart Grids

Eine Definition der Bundesnetzagentur [BNA11] beschreibt *Smart Grids* wie folgt: „Das konventionelle Elektrizitätsnetz wird zu einem Smart Grid, wenn es durch Kommunikations-, Mess-, Steuer-, Regel- und Automatisierungstechnik sowie IT-Komponenten aufgerüstet wird. Im Ergebnis bedeutet *smart*, dass Netzzustände in *Echtzeit* erfasst werden können und Möglichkeiten zur Steuerung und Regelung der Netze bestehen, so dass die bestehende Netzkapazität tatsächlich voll genutzt werden kann.“ Smart Grids sollen es demnach ermöglichen sowohl Erzeuger als auch Verbraucher in einem Netzwerk zu verbinden. Neben der reinen Übertragung des Stroms ist somit auch der Informationsaustausch zwischen den einzelnen Teilnehmern von großer Bedeutung. Zu diesen Teilnehmern gehören Stromerzeuger, -verbraucher, Speicheranlagen und IKT-Infrastrukturanbieter. Letztere haben laut

[KS+09] gute Voraussetzungen, Erfahrung und Mittel zur Bereitstellung der notwendigen Infrastruktur zum Austausch der Informationen und Daten.

### Smart Homes

Als *Smart Home* [Lip14a] werden Häuser bezeichnet, die mithilfe von Smart Metern den Stromverbrauch diverser Elektrogeräte automatisiert regeln. Dazu gehören unter anderem die Regelung der Beleuchtung, sowie der Heizung oder anderer Geräte. Im Vordergrund steht dabei die effiziente Nutzung des verfügbaren Stroms und somit eine gleichmäßigere Lastverteilung des Verbrauchs. Ein Beispielszenario ist das Laden eines Elektroautos: Wenn eine Person gegen 18:00 Uhr nach Hause kommt und sein Auto, wie viele andere auch, an das Stromnetz anschließt und es zum nächsten Morgen für den Weg zur Arbeit wieder aufladen möchte, kann mithilfe von Smart Metern das Smart Home entscheiden, wann das Auto geladen wird. Beispielsweise ist 02:00 Uhr nachts der Verbrauch eher gering, da dann die meisten Personen schlafen und ein Großteil der Stromverbraucher abgeschaltet ist. Bei einem flexiblen Stromtarif wäre der Strom zu diesem Zeitpunkt wahrscheinlich günstiger, da die Nachfrage gesunken ist. Daraus ergibt sich neben der Verteilung der Last eine potenzielle Kostenersparnis für den Kunden.

### Smart Meter

Smart Meter [Lip14b] sind Geräte, die sowohl die Durchflussmenge von Strom als auch die von Wasser und Wärme messen können. Dabei ist das Gerät in der Lage, den Durchfluss bidirektional zu protokollieren. Weiterhin sollen Smart Meter den Anwender dabei unterstützen Kosten zu reduzieren, indem der Stromverbrauch an den aktuellen Strompreis koppelt und beispielsweise das im Abschnitt 2.3 genannte Szenario automatisch regelt. Hierzu ist es jedoch notwendig, dass Stromanbieter solche flexible Tarife zur Verfügung stellen. Diese Geräte stellen die Schnittstelle zwischen den Stromerzeugern und Verbrauchern sowie dem intelligenten Stromnetz dar.

### 3.2.2. Systemarchitektur

Nachdem die obengenannten Begriffe von der konventionellen Energieversorgung abgegrenzt wurden, stellt sich die Frage, was ein Smart Grid eigentlich ausmacht. Um dieser Frage nachzugehen, beschäftigt sich dieser Abschnitt mit der Systemarchitektur von Smart Grids. Dies beinhaltet beispielsweise auch die Kommunikation und den Datenaustausch zwischen einzelnen Komponenten.

### Stromnetze als Smart Grid Komponenten

Um ein Stromnetz als Smart Grid bezeichnen zu können, muss es verschiedene Anforderungen erfüllen. Hierzu lassen sich laut Bundesnetzagentur in weiten Teilen die vorhandenen Übertragungsnetze aufrüsten, so dass unter Umständen nicht überall ein Netzausbau sondern lediglich eine Netzaufrüstung notwendig wird. Allerdings seien die Kapazitäten der Aufrüstung nicht ausreichend um den zusätzlichen Belastungen gerecht zu werden, weshalb es weiterhin nötig bleibt, konventionelle Übertragungsnetze weiter auszubauen [BNA11]. In Abbildung 3.7 ist zu sehen, dass die Übertragungsnetze eine zentrale Rolle spielen. Um diese steuern bzw. Informationen über Auslastung, Verfügbarkeit oder (aktuellen) Verbrauch erhalten zu können, werden Informations- und Kommunikationstechnologien benötigt, die mithilfe verschiedener Schnittstellen diese gewonnenen Informationen an die übrigen Netzteilnehmer übermittelt. Wichtig ist hierbei auch die Kommunikation der unterschiedlichen Teilnehmer untereinander, damit z.B. ein Verbraucher darüber informiert werden kann, dass von Erzeugerseite aktuell ein Energieüberschuss vorhanden ist.

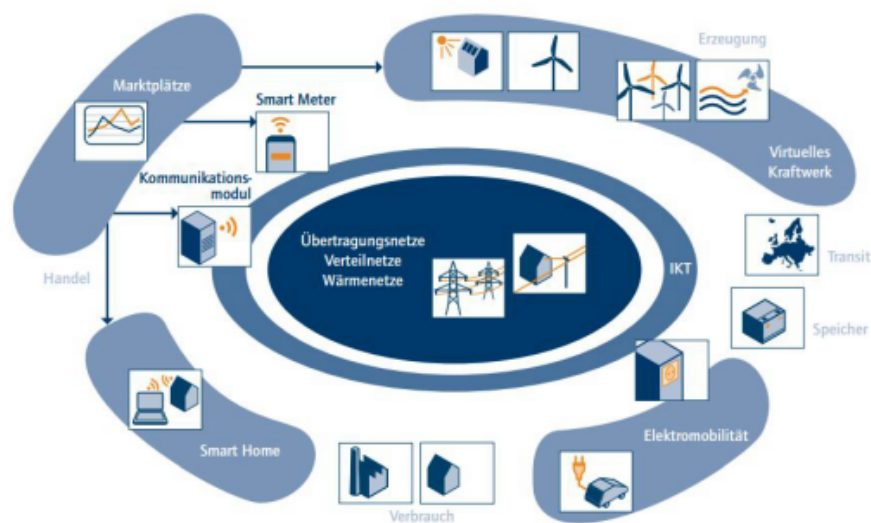


Abbildung 3.7.: Systemmodell des European Electricity Grid Initiative and Implementation plan [AK+12]

Das Netz selbst lässt sich in die Kernkomponenten *Transportnetz* und *Verteilernetz* unterteilen. Das Transportnetz dient der flächendeckenden Stromversorgung auf Hochspannungsebene und ist für den Transport des Stroms über große Distanzen verantwortlich. Der weitaus größere Teil unseres Stromnetzwerkes stellen die Verteilernetze dar. Sie leiten den Strom auf Mittel- und Niederspannungsebene vor Ort bis zum Verbraucher weiter und machen den Großteil des Gesamtnetzes aus.

Bislang bestand die Hauptaufgabe der Stromanbieter darin, darauf zu achten, dass zu jeder Zeit ausreichend Strom (in konstanter Spannung) vorhanden ist. Durch die zunehmende Dezentralisierung der Stromerzeugung im Zuge des Ausbaus der erneuerbaren Energien verändert sich auch die Nutzung des Stromnetzes, da immer mehr Privathaushalte oder Fabriken mit eigenen kleinen Kraftwerken zur Einspeisung des Stroms, vor allem auf Mittel- und Niederspannungsebene, beitragen. Smart Homes agieren also nicht nur als Verbraucher sondern auch als Erzeuger und Zwischenspeicher. Smart Grids sollen mithilfe von IKT ermitteln, wie man jederzeit Erzeugung und Verbrauch optimal regeln kann. Dazu gehören Informationen über die Netzauslastung und die verfügbaren Reservekapazitäten sowie den (aktuellen) Strompreis. Dadurch sollen Verbrauchsspitzen minimiert und eine möglichst konstante Nutzung des Stromnetzes geschaffen werden. Das führt wiederum zu einem stabileren Netz und beugt Stromschwankungen und gar Stromausfällen oder -abschaltungen aufgrund von Überlastungen vor.

### 3.2.3. Kommunikation

Wie in Abbildung 3.7 zu sehen ist, bieten verschiedene Schnittstellen die Möglichkeit über ein Kommunikationsnetzwerk das Übertragungsnetz mit dessen Teilnehmern zu verbinden und detaillierte Daten auszutauschen. Hierzu wird ein Kommunikationsnetz benötigt, welches aktuelle Informationen, Zustandsdaten und andere Messdaten des Netzwerkes übertragen kann. Die Fraunhofer-Einrichtung für Systeme der Kommunikationstechnik ESK hat als Teile der Smart Grid Kommunikationseinheiten folgende Aufgaben und Anwendungen genannt [FEK14]:

- Erweiterte (Strom-)Messinfrastruktur/ Smart Metering
- Lastmanagement/ Demand Side Management
- E-Mobilität/ Electric vehicles
- Dezentrale Energieerzeugung und Energiespeicherung/ Distributed Energy Resources and Storage
- Verteilnetzmanagement/ Distribution Grid Management

Dabei hängt je nach Anwendungsfall ab, welche Parteien untereinander was für Informationen kommunizieren können und wie verschiedene Aspekte im Bezug auf den Datentransfer zu regeln sind. Bei der Kommunikationsarchitektur sollen überwiegend offene Standards verwendet werden um die Entwicklung Anbieter-unabhängig und somit schneller zu gestalten. Damit diese Daten erhoben werden können hat der Bundestag im „Gesetz zur Öffnung des Messwesens bei Strom und Gas“



[DBUo8] vereinbart, dass seit 2010 bei Häusern, die neu an das Stromnetz angeschlossen oder grundlegend renoviert werden, Messeinrichtungen einzubauen sind, die es dem Anschlussnutzer ermöglichen jederzeit detaillierte Informationen über den Energieverbrauch und den Nutzungszeitraum ablesen zu können. In dem Projekt der Fraunhofer-Einrichtung wird für das Smart Metering eine Client-Server-Architektur verwendet, bei der eine zentrale Stelle im Haus die Daten der verschiedenen Verbrauchsgeräte aggregiert und Steuerbefehle ausführen kann, die der Benutzer über ein mobiles Endgerät einsehen, bzw. erteilen und ändern kann.

### Daten und Datenschutz

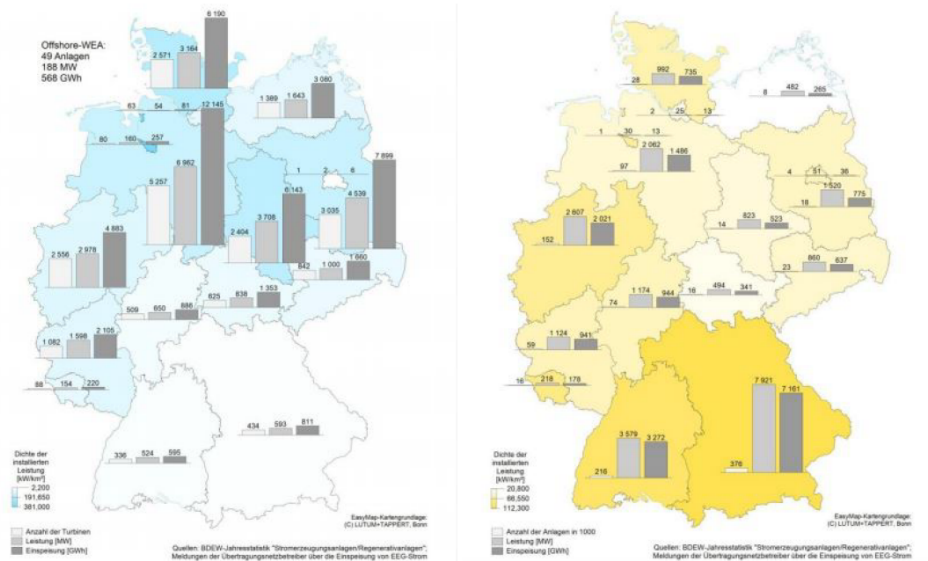
Beim Demand Side Management sollen Daten über beispielsweise das Ein- und Ausschalten der Stromversorgung, netzbetriebsrelevante Messdaten oder die simplen Zählerstände übertragen werden. Da für eine effiziente Nutzung des Smart Grids dieses in der Lage sein muss Stromverbraucher dynamisch ein- oder auszuschalten und daher auch Daten über den Betriebszustand im Netzwerk vorhanden sein müssen, hat die Konferenz der Datenschutzbeauftragten des Bundes und der Länder vom 27. Juni 2012 eine Orientierungshilfe zum datenschutzgerechten Smart Metering [BDI12] beschlossen, in der verschiedene Punkte zur Nutzung und Verwaltung der Daten im Bezug auf Smart Metering aufgefasst werden. Demnach sollen Ablesintervalle so groß gewählt werden, dass keinerlei Rückschlüsse über das Verhalten der Nutzer anhand des Verbrauchs gezogen werden können. Weiterhin sollen die Daten möglichst anonym und lediglich an jene Stellen übermittelt werden, die diese unbedingt benötigen. Ein weiterer Punkt ist die Datenhaltung. Dazu wird angemerkt, angemessene Löschfristen für die gespeicherten Daten festzulegen und dem Nutzer die vollständige Kontrolle über den Zugang zu Daten und Geräten zu sichern.

### 3.2.4. Herausforderungen

Wie bereits beschrieben, besteht das aktuelle Stromnetz aus großen Transportleitungen, die den Strom über große Distanzen mithilfe von Wechsel- bzw. Drehstromkabeln von den großen Kraftwerken über eine große Strecke verteilen. Die Verteilernetze sorgen dafür, dass dieser Strom, ausgehend von zahlreichen Kopplungspunkten flächendeckend bis hin zu den Verbrauchern transportiert wird.

Da Stromnetze generell nicht in der Lage sind Strom zu speichern wird der Strom in der Regel so produziert, dass der aktuelle Verbrauch gedeckt ist. Dabei kann es vor allem in lokalen Verteilernetzen zu Verbrauchsspitzen kommen, die das Netz stark belasten und bis zur zeitweiligen Abschaltung oder gar Ausfällen kommen kann. Im Folgenden sollen die sich ändernden Herausforderungen an das Stromnetz betrachtet werden und inwiefern sich die Gegebenheiten durch erneuerbare Energien und den Einsatz bzw. Ausbau von Smart Grids verändern können.





(a) Nutzung der Windenergie

(b) Nutzung der Photovoltaik

Abbildung 3.8.: Nutzungsverteilung der erneuerbaren Energien in Deutschland im Jahr 2011

### Erneuerbare Energien

Bei den erneuerbaren Energien haben sich vor allem die Photovoltaik und die Windkraft in den letzten Jahren stark entwickelt. Grundsätzlich sind diese Ressourcen überall abrufbar, jedoch lassen sich regionale Unterschiede in der Nutzungseffizienz feststellen. In Deutschland sind diese Unterschiede deutlich zu erkennen. Die Nutzung der Windkraft ist laut Bundesverband der Energie- und Wasserwirtschaft e.V. hauptsächlich in den Regionen Nord- und Ostdeutschlands weit verbreitet, wohingegen die Nutzung der Photovoltaik hauptsächlich im Süden stattfindet. Abbildung 3.8 [BEW13] zeigt graphisch die Verteilung der Nutzung, gemessen an der Anzahl der Anlagen, deren Leistung und Einspeisung ins Stromnetz. Die Auswirkungen auf das Verteilernetz werden im Abschnitt 3.2.4 betrachtet.

### Transportnetze

Das Transportnetz besteht im Wesentlichen aus Hochspannungswechsel- bzw. Drehstromkabeln [Pas14b], die Strom über große Distanzen transportieren können. Ein Vorteil dieses Verfahrens ist, dass Strom an Transformatoren umgespannt und für die Verbraucher nutzbar gemacht werden kann. Ein wesentlicher Nachteil besteht jedoch darin, dass auf größeren Distanzen von mehreren 100 Kilometern deutliche Verluste zu verzeichnen sind. Außerdem erzeugen solche Kabel ein elektromagnetisches Feld, welches Einfluss auf das direkte Umfeld dieser Kabel hat, wodurch diese

bspw. für den maritimen Gebrauch eher ungeeignet sind. Durch die vermehrte Nutzung erneuerbarer Energien wird die Stromerzeugung und somit auch der Transport zunehmend dezentraler, wodurch sich die Anforderungen an das Stromnetz ändern. Für Transportnetze sind besonders die Offshore-Windparkanlagen eine besondere Herausforderung, da die Kabel hier nicht als Erdkabel oder Freileitungen verlegt werden können. Hierfür wurden spezielle Gleichstrom-Seekabel entwickelt, die den enormen Vorteil mit sich bringen, Strom auch über mehrere Tausend Kilometer relativ verlustfrei transportieren zu können. Allerdings muss dieser Strom wieder in Wechselstrom umgewandelt werden, damit Verbraucher ihn nutzen können. Dazu wird bei Offshore-Anlagen der Gleichstromtransport von der Erzeugerstation bis zum Land vorgenommen, wo er daraufhin transformiert und weitergeleitet wird. Ein weiterer Aspekt dieser Art der Stromübertragung ist, dass bei Gleichstrom lediglich eine Punkt-zu-Punkt Verbindung möglich ist, wodurch die Leitung nicht an verschiedenen Zwischenpunkten angezapft werden kann. Deshalb wurde diese Methode bislang im inländischen Stromnetz nicht verwendet.

### Verteilernetze

Verteilernetze sind besonders von den sich ändernden Gegebenheiten betroffen. Nicht nur, dass der Strom bidirektional fließt, sondern auch und gerade die Lastspitzen führen das Netz an seine Grenzen, da der Stromverbrauch stetig steigt [Cle13]. Smart Grids sollen auch dazu beitragen, diese Auslastung zu regulieren, da durch die smarte Steuerung von Elektrogeräten Lastspitzen verringert werden können. Bei nicht zeitkritischen Aktionen, wie dem Aufladen des Elektroautos, dem Waschen der Wäsche oder der Regulierung der Temperatur im Gefrierschrank können Smart Meter den Verbrauch auf einen Zeitpunkt verschieben, an dem der Strom gerade günstig, bzw. das Netz weniger belastet ist. Dadurch profitiert das Verteilernetz von geringeren Spitzenlasten was insgesamt die Netzstabilität steigert. Dazu muss das Netz möglichst effizient aufgerüstet werden, damit solche Automatisierungen möglich werden.

### 3.2.5. Ausblick

Smart Grids haben ein großes Potenzial die Stromversorgung zu sichern und zu stabilisieren. Sie ermöglichen es, erneuerbare Energien effizient zu nutzen und bieten somit die Grundlage vieler neuer Technologien. Durch den Ausbau der Netze kann eine Infrastruktur geschaffen werden, die es in Zukunft ermöglicht und deutlich vereinfacht beispielsweise die Elektromobilität voran zu treiben. Auch die intelligente Vernetzung von Häusern und deren Elektrogeräte bieten Spielraum für Innovationen. RWE hat bereits Konzepte für ein Smart Home entwickelt, in dem die Vernetzung über das Smart Grid hinaus geht und Geräte auch über das Internet oder das

Mobilfunknetz Informationen für den Benutzer bereitstellen. Allerdings birgt diese Technologie auch einige Gefahren und stößt nicht überall auf uneingeschränktes Wohlwollen. Bei einem derart verzweigten Netz aus Energie und Information bietet es auch Schwachstellen für Angreifer, die theoretisch bei einem erfolgreichen Angriff auf das Steuerungssystem große Teile der Versorgung beeinflussen können. Auch ist zu beachten, dass es trotz der potenziellen Stromersparnisse als störend empfunden werden kann, wenn nachts die Waschmaschine schleudert oder die Geschirrspülmaschine Wasser pumpt. Die Protokollierung und Bereitstellung der Betriebszustände von Geräten lässt auch einen Rückschluss auf das Verhalten der Bewohner eines Hauses ziehen. Der Gesetzgeber versucht jedoch für solche Themen Vorschläge und Regelungen zu finden [BDI12], damit die Entwicklung dieser Technologie und somit deren Ausbau weiter an Schwung gewinnt und zukünftig eine Energieversorgung möglich wird, die sich weitgehend auf erneuerbare Energien stützt und in Kooperation mit anderen Nationen dazu beiträgt, die Verwendung fossiler Brennstoffe zu minimieren.

## 4. Anforderungen

Für das Programm *Maverig* wurden stetig Anforderungen in Absprache mit der Betreuung, die als Auftraggeber agiert, spezifiziert oder neu definiert. Dies hatte zum Einen damit zutun, dass im Laufe des Projekts mehr Wert seitens der Betreuer auf die Stabilität und Zuverlässigkeit der Software als auf optionale Kann-Anforderungen gelegt wurde, wie beispielsweise eine 3D-Visualisierung der Simulation. Zum Anderen ergaben sich aus dem guten Projektfortschritt und aus den beiden durchgeführten Usability-Studien neue Anforderungen, die vorher nicht eingeplant waren, wie beispielsweise das dynamische Hinzufügen von Komponenten im Komponenten-Assistenten.

### 4.1. Funktionale Anforderungen

Im Folgenden werden die ursprünglichen Anforderungskataloge der Komposition und Simulation im Vergleich zu den Endanforderungskatalogen dargestellt. Die fett geschriebenen Textstellen sind die Anforderungen, die im Laufe des Projekts zusätzlich entstanden oder genauer spezifiziert worden sind. Die normal formatierten Textstellen sind die Anforderungen, die bereits am Anfang des Projekts festgesetzt wurden. Die hellgrauen Textstellen sind Anforderungen, die als Kann-Anforderungen deklariert wurden und somit nicht zwingend umgesetzt werden mussten. Um eine Einhaltung und Erweiterung der funktionalen Anforderungen belegen zu können, befindet sich im Anhang neben dem Benutzerhandbuch eine Auflistung des Funktionsumfangs wieder.

#### **Funktionale Anforderungen an die Komposition:**

1. Es muss ein Fenster mit einer Menüleiste geben.
2. Es muss eine Toolbar geben, über die man Einstellungen vornehmen kann.
3. Es muss ein Moduspanel geben, in dem Icons für die Komponenten gruppiert werden können.
  - a) **Icon bei Auswahl hervorheben (größeres Icon, Schatten, etc.) im Komponenten-Modus.**

**b) Kein Icon ausgewählt (Transparenz, Icons kleiner) im Selektions-Modus.**

4. Es muss ein Eigenschaftenpanel mit generischer Property-Value-Tabelle geben.
5. Es muss ein 2D-Canvas für den visuellen Szenarioeditor geben.
6. Es muss eine ausblendbare Log-Ausgabe geben.
7. Das Programm muss für Internationalisierung (Standard: Englisch) ausgelegt sein.
8. **Es muss ein Hilfe-Menü mit About-Dialog geben.**
9. Icons und Knotenpunkte müssen gezeichnet werden können.
10. Icons müssen selektiert werden können.
11. Selektierte Icons müssen verschoben werden können.
12. Linien und Linie-Icon-Linie Konstrukte müssen gezeichnet werden können.
13. Eine Linie muss selektiert werden können.
14. **Linienendpunkte müssen bei Selektion angezeigt werden.**
15. **Linien und Endpunkte müssen verschoben werden können, wobei Linienerverbindungen zu anderen Elementen beibehalten werden müssen.**
16. Komponenten müssen per Drag& Drop vom Moduspanel in das Szenario gezeichnet werden können.
17. Elemente müssen per Drag&Drop vom Moduspanel in das Szenario gezeichnet werden können.
18. Durch die „Strg“-Taste müssen mehrere Elemente selektiert werden können.
19. Elemente müssen durch eine Auswahlbox selektiert werden können.
20. **Aktive Elemente müssen durch Strg+A selektieren werden können (Modus abhängig).**
21. **Alle Elemente eines Icon- bzw. Linien-Typs müssen per Doppelklick selektiert werden können.**
22. Icons müssen aneinander andocken können.

23. Es muss einen gepunkteten Verbindungspfeil für verbundene Icons geben.
24. Linienendpunkte müssen an Icons an- und abdocken können.
25. Elemente müssen beim Verschieben andockt bleiben.
26. Selektierte Elemente müssen entfernbar sein (Entf-Taste, Toolbar-Button).
27. Selektierte Elemente müssen kopiert werden können (Strg+C).
28. Elemente müssen aus der Zwischenablage eingefügt werden können (Strg+V).
29. Selektierte Elemente müssen ausgeschnitten (Strg+X) werden können.
30. Selektierte Elemente müssen verdoppelt werden können.
31. Das Szenario muss vergrößert werden können.
32. Das Szenario muss verkleinert werden können.
  - a) **Szenario in Zeichenfläche einfangen wenn Szenario größer als Zeichenfläche.**
  - b) **Szenario zentrieren wenn Szenario kleiner als Zeichenfläche.**
33. Das Szenario muss mit dem Mausrad (Zoomrichtung Mausposition) vergrößert und verkleinert werden können.
34. **Das Szenario muss auf Panel-Größe angepasst werden können (Toolbutton, Zoom-to-fit Szenario).**
35. Die Szenarioposition muss verschoben werden können (Hand-Modus).
36. **Es muss ein Raster zum Ausrichten der Elemente geben.**
37. **Elemente müssen an Raster-Kreuzpunkte andocken.**
38. **ForceAtlas 2 muss für Graphenoptimierung eingebunden werden.**
39. **Auto-Layout mit ForceAtlas 2 muss vorhanden sein.**
40. Komponentenkonfiguration für Basis-Komponenten müssen angelegt werden – Eigenschaften:
  - a) Pfad zur Simulatorimplementierung
  - b) Kategorienname für die Gruppierung im Moduspanel
  - c) Icon-Symbol

- d) Darstellungsform als Icon, Linie oder Linie-Icon-Linie
  - e) Attribute sowie deren Typbezeichnung, Einheiten und Referenzwerte
  - f) Erlaubte Verbindungen zu anderen Komponenten und verbundene Attribute
41. Szenariomodell muss verwaltet werden können – Eigenschaften:
    - a) Simulationsstartzeitpunkt und -dauer
    - b) **Komponenten-, Selektions- und Hand-Modus**
    - c) Komponenteninstanzen und deren Parameterwerte und Positionen
    - d) Verbindungen
    - e) **Selektierte Elemente**
    - f) **Events für Modell-Änderungen**
  42. Komponenten aus der Konfiguration (Basis-Komponenten) müssen im Moduspanel angezeigt werden.
  43. **Komponenteninstanzen müssen nach create/ delete mit Modell synchronisiert werden.**
  44. **Komponenteninstanzen müssen nach copy& paste verwaltet werden.**
  45. **Komponenten-, Selektions- und Hand-Modus muss vorhanden sein.**
  46. **Parameter müssen mit Modell synchronisiert werden.**
  47. Parameter eines selektierten Elements müssen im Eigenschaftenpanel bearbeitet werden können.
  48. Parameter mehrerer selektierter Elemente gleichen Typs müssen bearbeitet werden können.
  49. Es muss eine Visualisierung des Online/ Offline-Modus von Trafos und Linien geben.
  50. Selektierte Elemente müssen mit Modell synchronisiert werden.
  51. Verbindungen müssen mit Modell synchronisiert werden.
  52. Es muss eine Validierung der Verbindungen beim Andocken von Elementen geben (dezentale Fehlermeldung).
  53. Es muss der Simulations-Startzeitpunkt und die Dauer festgelegt werden können.

54. Das Szenario muss gespeichert werden können.
55. Das Szenario muss geladen werden können.
56. **Es muss eine Szenario-Historie geben.**
57. **Es muss eine Rückgängig- und Wiederherstellen-Funktion geben.**
58. **Es muss ein Simulator für CSV-Dateien anstelle von HouseholdSim für Häuser implementiert werden.**
59. **Es müssen (CSVs, Parameter etc.) für ein Demi-Szenario gesammelt werden.**
60. **Es muss eine Simulationsmethode bereitgestellt werden und diese muss in einem separaten Prozess gestartet werden können.**
61. **Die Mosaik-Ausgaben müssen im Log angezeigt werden.**
62. **Komponenten müssen auf Simulator-Implementierung gemappt werden.**
63. **RefBus, Bus, Branch, Transformer müssen in PyPower-JSON gespeichert werden.**
64. **Modellinstanzen müssen erzeugt und Referenzen in Komponenteninstanzen gespeichert werden können.**
65. **Verbindungen müssen umgesetzt werden.**
66. **Mosaik-Webvisualisierung muss eingebunden und angezeigt werden.**

#### **Funktionale Anforderungen an die Simulation:**

1. Die Komposition und Simulation müssen auf Modi-Wechsel reagieren können.
  - a) **Play: Simulationpanel anzeigen**
  - b) **Stop: Kompositionpanel anzeigen**
  - c) **Restart: Im Simulationsmodus bleiben**
  - d) **Überflüssige Buttons verstecken**
2. Es muss skalierbare Echtzeitdiagramme geben.



3. Es muss ein Attributepanel mit Echtzeitdiagrammen geben.
4. **Es muss eine Simulationsfortschrittsleiste geben.**
5. **Alle Texte müssen übersetzt werden (Standard: Englisch).**
6. **Es muss ein Benutzerhandbuch im Hilfe-Menü geben.**
7. **Es muss eine Geschwindigkeitsoptimierung geben für:**
  - a) ForceAtlas 2 (Cython, BarnesHut, AutoSpeed)
  - b) Validierung, Snap-Zone (Cython, BarnesHut)
  - c) Positionierung beim Bewegen vieler Elemente
  - d) **Selektieren vieler Elemente**
8. Es muss ein Visualisierungssimulator zur dynamischen Datenabfrage implementiert werden.
9. Der Simulationsfortschritt muss im Modell gespeichert werden.
10. Die Attributwerte vom Visualisierungssimulator müssen empfangen und bei den entsprechenden Komponenteninstanzen gesetzt werden.
11. **Referenzwerte (min, max) im Komponentenmodell müssen gesetzt werden.**
12. **Eine Normierung/Skala muss überlegt werden.**
13. **Es müssen negative und positive Stromrichtungen berücksichtigt werden.**
14. Es müssen Farbbereiche für passende Skalen festgelegt werden.
15. Farbwert-Interpolation und aktuelle Heatfarbwerte müssen berechnet werden.
16. Die Simulationsfortschrittsleiste muss sich aktualisieren.
17. Das Attributepanel muss sich mit Echtzeitdaten von selektieren Elementen aktualisieren.
18. Heat-Einfärbungen für View-Elemente müssen aktualisiert werden.
19. **Heat-Einfärbungen müssen durch verschiedene Konzepte, wie Balken und Schatten, visualisiert werden.**
20. **Es muss eine Tag-/Nachtvisualisierung in der Simulation geben.**

21. **Der Play-Button in der Toolbar muss während der Simulation zum Pause-Button werden.**
22. **Die Uhrzeit muss links/rechts neben der Thimeshift-Leiste angezeigt werden.**
23. Beim Drücken vom Pause-Button muss die Simulation im Hintergrund weiterlaufen.
24. **Simulationszeitpunkt muss gesetzt und pausiert werden.**
25. **Vergangene Simulationsdaten müssen eingespielt werden können.**
26. Es soll eine 3D-Anbindung auf GraphicsView mit Minimal-3D-Beispiel geben.
27. Es soll eine 3D-Welt mit leerer Landschaftsebene gezeichnet werden.
28. Es sollen 3D-Modelle (Standalone) für PV, WKA, Trafo und Netzknoten vorhanden sein.
29. 3D-Modelle wie Kabel und Leitungen sollen skalierbar sein.
30. Es soll 3D-Modelle für Haus, BHK, Dach-PV und EV geben.
31. Es soll 3D-Modelle für verschiedene Hausgrößen (Mehrfamilienhäuser) geben.
32. Es soll eine Kamerasteuerung für die Maus implementiert werden (isometrisch oder frei bewegbar).
33. Die Toolbar-Zoombuttons sollen auf Kamerasteuerung angewendet werden.
34. Die 3D-Szene soll auf Basis des Szenariomodells (Demo) angelegt werden.
35. 3D-Modelle sollen einfärbbar sein.
36. 3D-Modelle sollen selektierbar sein.
37. Multiselect (Strg, Doppelklick und Strg+A) für 3D-Modelle soll möglich sein.
38. **Erweiterungsmöglichkeiten müssen gut und verständlich dokumentiert werden.**
  - a) **Ein Konzept für möglichst einfache Erweiterbarkeit und Kontrollstrategien sollen angefertigt werden.**
39. **Das Szenario-Modell muss editierbar gemacht werden (JSON Export+Import).**



40. **Es muss ein Mockup für den Komponentenassistenten erstellt werden.**
41. **Die GUI für den Komponentenassistenten muss vorhanden sein.**
42. **Haussymbole je nach Anzahl der Haushalte sollen anders dargestellt werden.**
43. **Es muss ein Konzept für mehrfaches Erstellen und Verbinden von Elementen erstellt werden.**

## 4.2. Nichtfunktionale Anforderungen

Im Rahmen der Projektgruppe dient die internationale Norm ISO/IEC 9126, die auch in der Norm ISO/IEC 25000 implementiert wurde, als ein Modell für nichtfunktionale Anforderungen. Die Norm bildet mit sechs übergeordneten Qualitätsmerkmalen und 26 untergliederten Teilmerkmalen ein Konstrukt, das auf sämtliche Softwarearten anwendbar ist. Die sechs übergeordneten Qualitätsmerkmale und somit Bereiche der Norm sind Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz und Wartbarkeit. Die Bedienung der Maverig Anwendung erfolgt über eine Desktop-Softwareanwendung, die allgemein verwendbar sein sollte. Das bedeutet, dass die gängigen Betriebssysteme Windows, Linux und Mac OS X unterstützt werden. Die Voraussetzung zum Betrieb der Softwareanwendung ist das Vorhandensein von *Python*. Es wird eine Mindestauflösung von 1024x768 unterstützt.

## 5. Systementwurf

### 5.1. Zweck des Systems

Der Zweck des Systems besteht darin, das Framework *Mosaik* um eine grafische Bedienoberfläche zu erweitern. Die Bedienoberfläche soll das Erstellen von Smart Grid Szenarien ermöglichen, wodurch Nutzer nicht wie bisher ein eigenes Python-Script schreiben müssen. Die anschließende Simulation der erstellten Szenarien soll ebenfalls über die Bedienoberfläche erfolgen. Damit soll die Nutzung von Mosaik vereinfacht und das Framework einer breiteren Masse an Nutzern zugänglich gemacht werden. Jedoch soll das System keinesfalls die bisherigen Möglichkeiten der Szenarienerstellung ersetzen, da sich umfangreiche, sowie komplexe Szenarien besser über ein Python-Script realisieren lassen. Vorrangiger Zweck des Systems ist deshalb die Erstellung kompakter sowie einfacher Szenarien und die Steuerung sowie Visualisierung der Simulation über eine homogene Bedienoberfläche.

### 5.2. Anwendungsfalldiagramm

Zu Beginn der Projektgruppe haben wir uns dem Thema genähert, indem wir intensiv über ein Anwendungsfalldiagramm diskutiert und es entworfen haben. Hierbei hat sich ein Ergebnis und eine Vorstellung bei allen Projektbeteiligten abgezeichnet, welche Funktionen unsere Anwendung realisieren soll.

Es existiert ein System mit dem Namen *Maverig*. Dieses System besitzt vier übergeordnete Funktionen. Mit Hilfe von *Maverig* können Anwender ein Smart Grid Szenario erstellen. Dieses Szenario kann mit diversen Parametern und Einstellungen bearbeitet werden. Es ist möglich, das Szenario in einer Simulation zu steuern. Hierbei nutzt das System *Maverig* das externe System *Mosaik*, um die Daten zu simulieren. *Mosaik* liefert konkrete Simulationsdaten für das zuvor erstellte Szenario. Es ist ebenfalls möglich eine Simulation zu analysieren und sich die Daten der einzelnen Komponenten anzeigen zu lassen. Hierfür wird ebenfalls das externe System *Mosaik* genutzt.

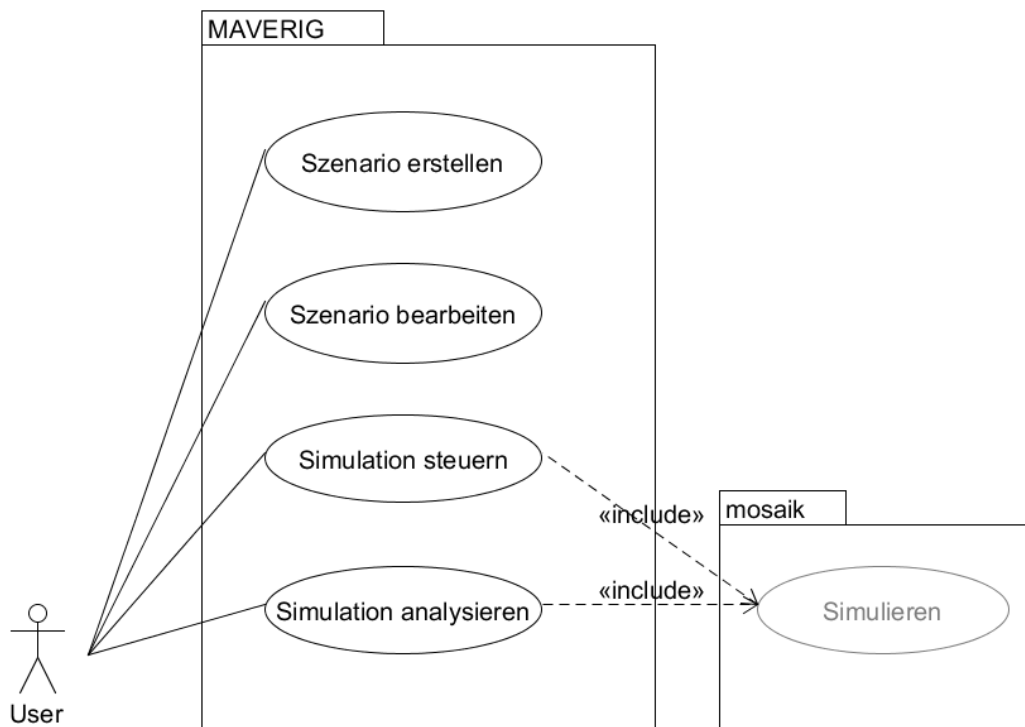


Abbildung 5.1.: Anwendungsfalldiagramm Maverig

### 5.3. Entwurfsziele

Unabhängig von der verwendeten Programmiersprache oder einer konkreten Implementierung, werden Ziele und Qualitätskriterien für den Entwurf festgelegt. Neben einer angemessenen *Funktionalität*, die den Anforderungen entsprechen sollte, wird das zu implementierende System *robust* entwickelt. Das heißt insbesondere, dass fehlerhafte Eingaben - sofern möglich - unterbunden werden, sich das System mit seinen Daten immer in einem definierten Zustand befindet und aussagekräftige Fehlermeldungen angezeigt werden.

Hinsichtlich der Installation, Bedienung und Anpassung der Software durch den Endbenutzer ist *Benutzerfreundlichkeit* ein weiteres Entwurfsziel. Das System muss eine möglichst einfach benutzbare Schnittstelle zur Bedienung bieten, die außerdem eine gute *Erlernbarkeit* gewährleistet. Das System soll *anpassbar* und einfach *erweiterbar* sein. Ein Beispiel für *Erweiterbarkeit* ist das nachträgliche Hinzufügen von Simulatoren.

Da das System unter anderem zu Präsentationszwecken eingesetzt wird, soll die *Benutzbarkeit* des Systems vor allem mit dem Fokus auf Zufriedenstellung des Nut-

zers berücksichtigt werden. Qualitätskriterien an den Programmcode sind vor allem *Lesbarkeit* und eine nicht zu aufwändige *Modifizierbarkeit*.

## 5.4. Einsatz von 2D und Einblick in 3D

Für die 2D-Oberfläche fiel die Entscheidung auf das Entwicklungsframework *Qt*. *Qt* ist ein leistungsstarkes Entwicklungsframework zur GUI-Programmierung, das weit verbreitet sowie gut dokumentiert ist und viele zusätzliche Funktionen liefert. Zur Nutzung des Entwicklungsframework in Python existieren die sich sehr ähnelnden Python-Bindings *PyQt* und *PySide*. *PyQt* unterstützt zwar eine höhere Version von *Qt*, ist aber nur mit einer GPL-Lizensierung verfügbar. *PySide* dagegen ist unter der LGPL-Lizenz veröffentlicht. Da *Mosaik* unter der LGPL veröffentlicht wurde und für die Software *Maverig* eine möglichst offene Lizenz angestrebt wird, verwenden wir *PySide* für die Entwicklung der Benutzeroberfläche.

Für den Bereich der Simulation war ursprünglich eine 3D-Modellierung angedacht. Der Benutzer bekommt dadurch einen besseren Einblick der Szene und erhält die Möglichkeit, in der virtuellen Welt mitzuwirken. Aus Zeitgründen haben wir uns gegen eine 3D-Modellierung entschieden und den Fokus auf eine voll funktionsfähige 2D-Umgebung bis zum Projektende im März 2015 gelegt. Die 3D-Modellierung bedarf einen hohen Zeitaufwand in die Einarbeitung der Software *Blender3D*, einer Game-Engine und der Berücksichtigung aller Aspekte der Ausführung der 3D-Szene und ihrer Objekte. *Blender3D* ist ein 3D-Modellierungs- und Animationswerkzeug, das Funktionalitäten wie Modellierung, Texturierung, Beleuchtung, Animation oder Videonachbearbeitung anbietet. Für die Umsetzung der 3D-Anwendung werden zunächst mit Hilfe von *Blender3D* verschiedene 3D-Objekte erstellt, wie die einzelnen Komponenten aus der Komposition sowie einer 3D-Szene mit einer leeren Landschaftsebene. Zur Modellierung können verschiedene Techniken eingesetzt werden, wobei die gängigste Art *Meshes* sind. Sie bestehen aus einzelnen Punkten, die durch Kanten miteinander verbunden werden, die wiederum Flächen aufspannen können. Mesh-Objekte bieten die größte Flexibilität beim Bearbeiten, Texturieren und Animieren. Nach dem Erstellen der 3D-Modelle können diese mit Hilfe der Game-Engine über die entsprechende API im Projekt visualisiert werden. Dazu werden die einzelnen Modelle, die für die Simulation benötigt werden, an den entsprechenden Stellen und der entsprechenden Skalierung der 3D-Szene eingebunden. Zusätzlich zu der Positionierung der Objekte ist es wichtig die Kamera auszurichten, die für die Betrachtungsweise der Objekte zuständig ist. Nach der Zusammenstellung der einzelnen Objekte und der Kamera können weitere grafische Effekte eingebunden werden, die das Aussehen der Bilder verändern. Zu diesen Effekten gehören unter anderem Oberflächentexturen, Licht- und Schatteneffekte und die Transparenz der Objekte. Interaktionen mit der Tastatur, Maus oder einem Joystick sind ebenfalls

möglich. Alles in Allem wären dies die Schritte, die durchlaufen werden müssten, um eine angemessene simple 3D-Umgebung zu realisieren. Ein weiterer Grund weshalb die 3D-Modellierung in Verzug geraten ist, ist die Auswahl der Game-Engine. Es gibt acht Game-Engines, die mit Python interagieren können. Hierbei sind zwei (PySoy und Ren'Py) nicht für diese Anwendung geeignet und drei (PyOgre, Panda3D und Blender3D) nicht mit Python 3.4 kompatibel. Eine weitere Einschränkung ist die Lizenz LGPL, der sich die Projektgruppe unterzogen hat. Dadurch fallen zwei weitere Game-Engines (PyGame und Soya3D) weg, weshalb die Game-Engine Pyglet momentan als einzige Möglichkeit bleibt. Pyglet ist eine plattformübergreifende Multimediabibliothek für Python. Sie bietet eine objektorientierte Programmierschnittstelle für die Entwicklung von Spielen und anderen grafikspezifischen Anwendungen. Ein Kritikpunkt an Pyglet ist das ausschließliche Arbeiten mit einer OpenGL-Schnittstelle und Grafiken, die über Vertices programmiert werden müssen. Es gibt bisher keine Möglichkeit 3D-Objekte einzubinden und diese weiterzuverarbeiten.

## 5.5. Zerlegung des Systems: Model-View-Presenter

Für die Implementierung des Systems wurde entschieden, das Entwurfsmuster *Model-View-Presenter* (MVP) zu verwenden. Das Entwurfsmuster beschreibt den Ansatz, das Modell (engl. Model) von der Ansicht (engl. View) zu trennen und über einen Präsenter (engl. Presenter) zu verbinden. Der Vorteil durch die strengere Trennung ist vor allem die verbesserte Testbarkeit und eine verbesserte Struktur des Systems. Für das Testen können die Views einfach durch Mock-Ups ersetzt werden.

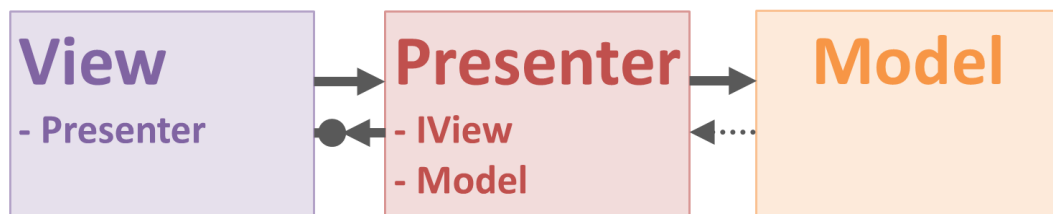


Abbildung 5.2.: Model-View-Presenter

Abbildung 5.2 veranschaulicht das MVP-Architekturmuster. Die *View* kennt den *Presenter*, der *Presenter* greift über ein Interface auf die *View* zu und kennt das *Model*. Das *Model* ist komplett unabhängig von *Presenter* und *View*.

Die erste Initialisierung findet im *Model* statt. Das *Model* registriert alle für die Anwendung notwendigen Daten, wie die Historie der durchgeführten Aktionen, aller gespeicherten Elemente inklusive der Parameter, Informationen über momentan angewandte Arbeitsmodi, Simulationseinstellungen und weiterer Daten. Eben-

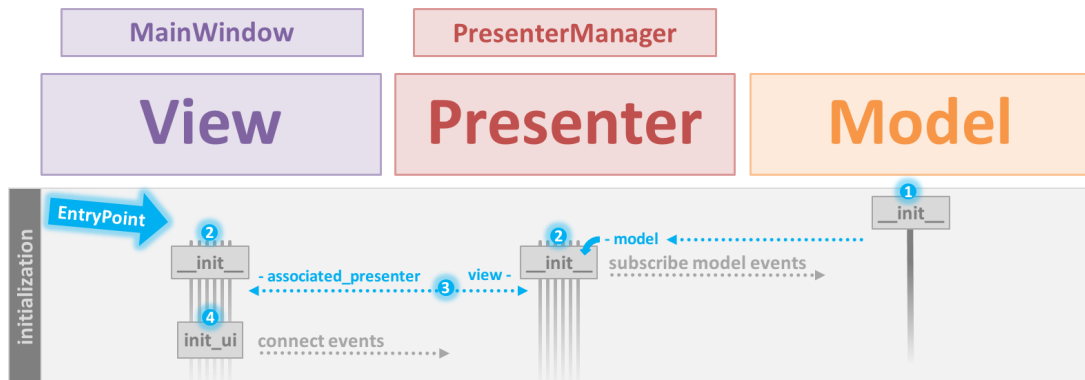


Abbildung 5.3.: MVP Initialisierung

so werden alle notwendigen Events initialisiert. Events sorgen für eine Reaktion der Anwendung, falls bestimmte Ereignisse auftreten.

Für alle notwendigen Presenter sowie alle korrespondierenden Views werden die Klassen PresenterManager und MainWindow benutzt. Der PresenterManager initialisiert alle untergeordneten Presenter. Bei der Erstellung dieser wird die Oberklasse, sowie das soeben erstellte Model referenziert. Parallel verwaltet das Hauptfenster die Erstellung untergeordneter Views und die Übergabe von sich selbst.

Bei der Erstellung von Presenter und View geschieht zugleich die Assoziation betreffender Komponenten aufeinander. Dies beinhaltet die Zuordnung von jeweils mit der View verbundenen Events zum Presenter. Dieser verzeichnet auftretende Events weiterleitend im Model.

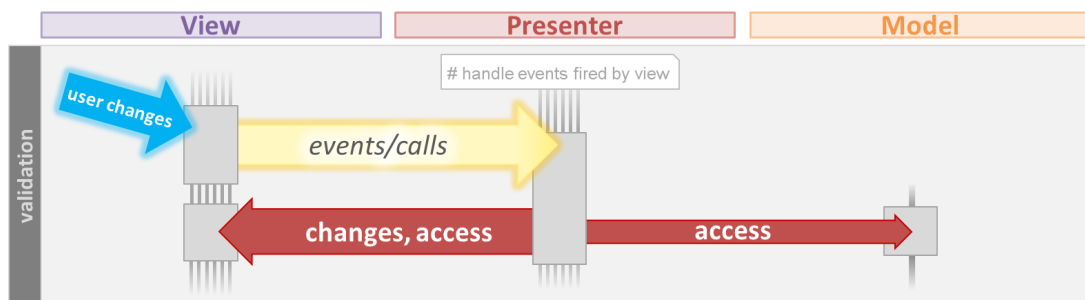


Abbildung 5.4.: MVP Validierung

Sind alle Events registriert, kann durch eine Änderung oder Aktion des Benutzers die betroffene View einen Event-Call im assoziierten Presenter aufrufen. Dieser veranlasst daraufhin rückwirkend eine Änderung der Ansicht der betreffenden View, sowie die Aktualisierung von Daten im Model.



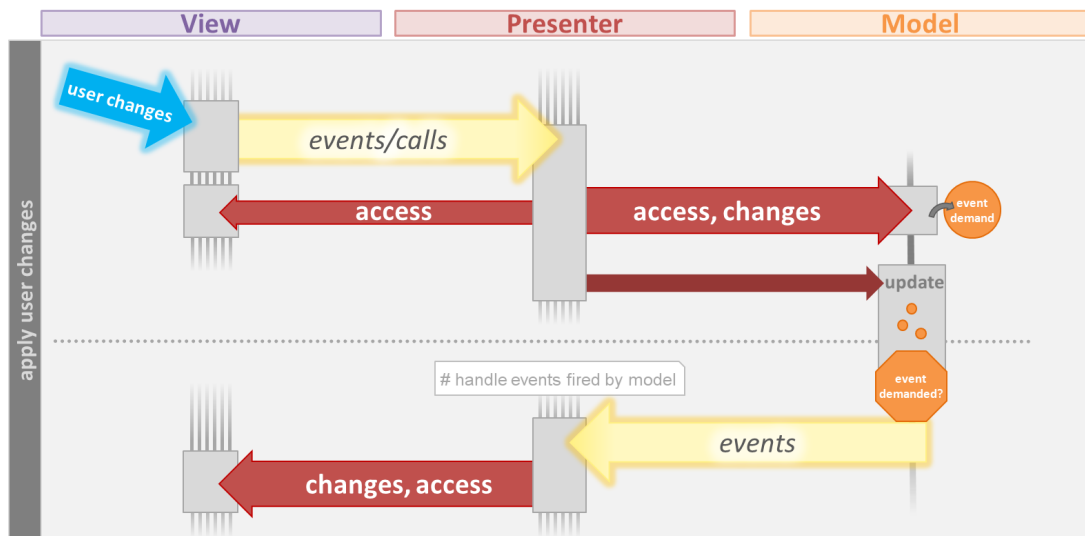


Abbildung 5.5.: MVP Verarbeitung von Benutzereingaben

## 5.6. Mock-Ups

Im folgenden Kapitel sind die Mock-Ups zu sehen, die wir für unsere verschiedenen Benutzeroberflächen entworfen haben.

### 1. Mock-Ups Gui

Da unsere Anwendung in die zwei großen Pakete *Komposition* und *Simulation* unterteilt wurde, haben wir für beide Bereiche Mock-Ups erstellt. In 5.6 ist das Mock-Up für die Komposition zu sehen. Die nachfolgende Grafik 5.7 zeigt das Mock-Up für die Simulation, das Ähnlichkeiten in der Struktur und Bedienung aufweist. Allerdings sind die linke Menübar und die Inhaltsbox in der Mitte anders gefüllt.

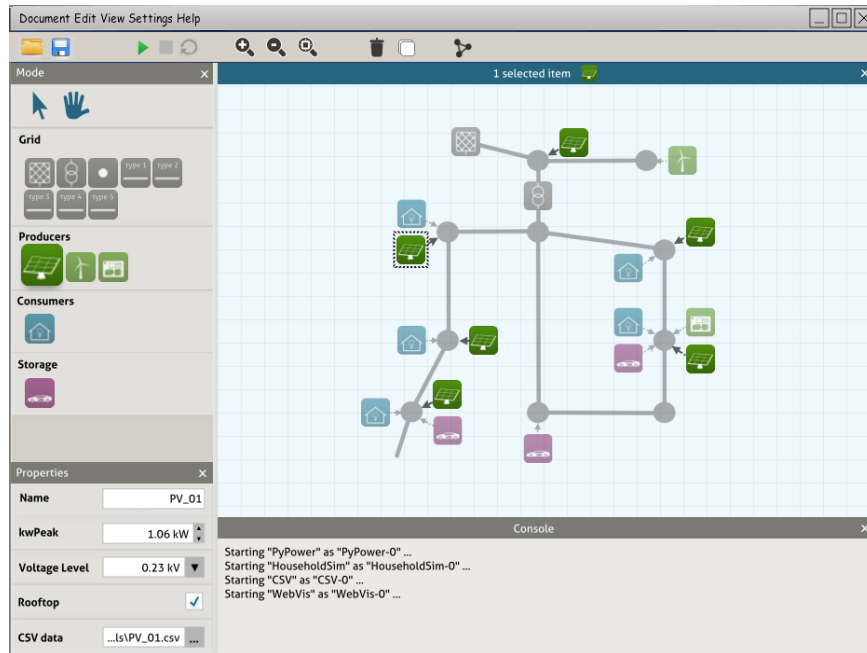


Abbildung 5.6.: Mock-Up der Kompositionsansicht

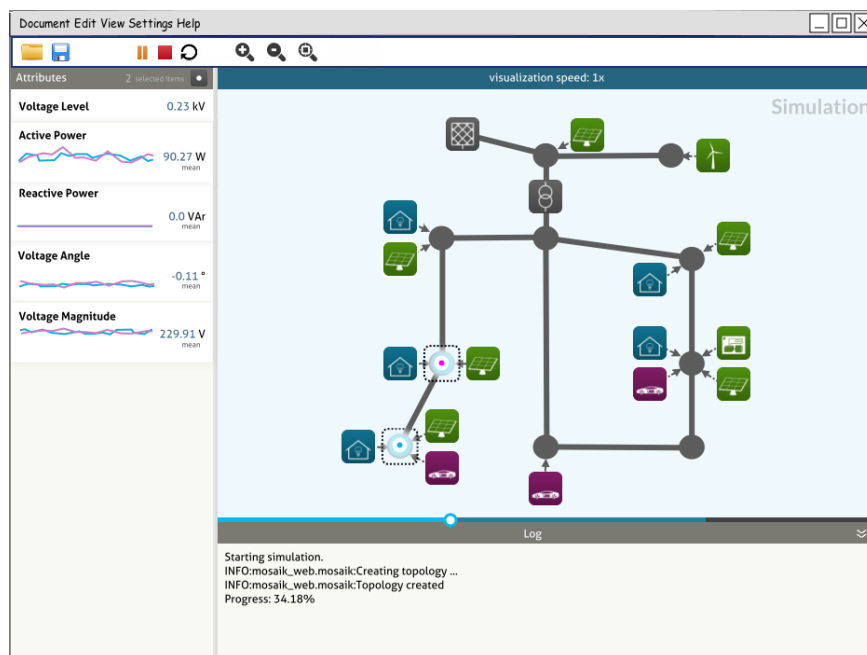
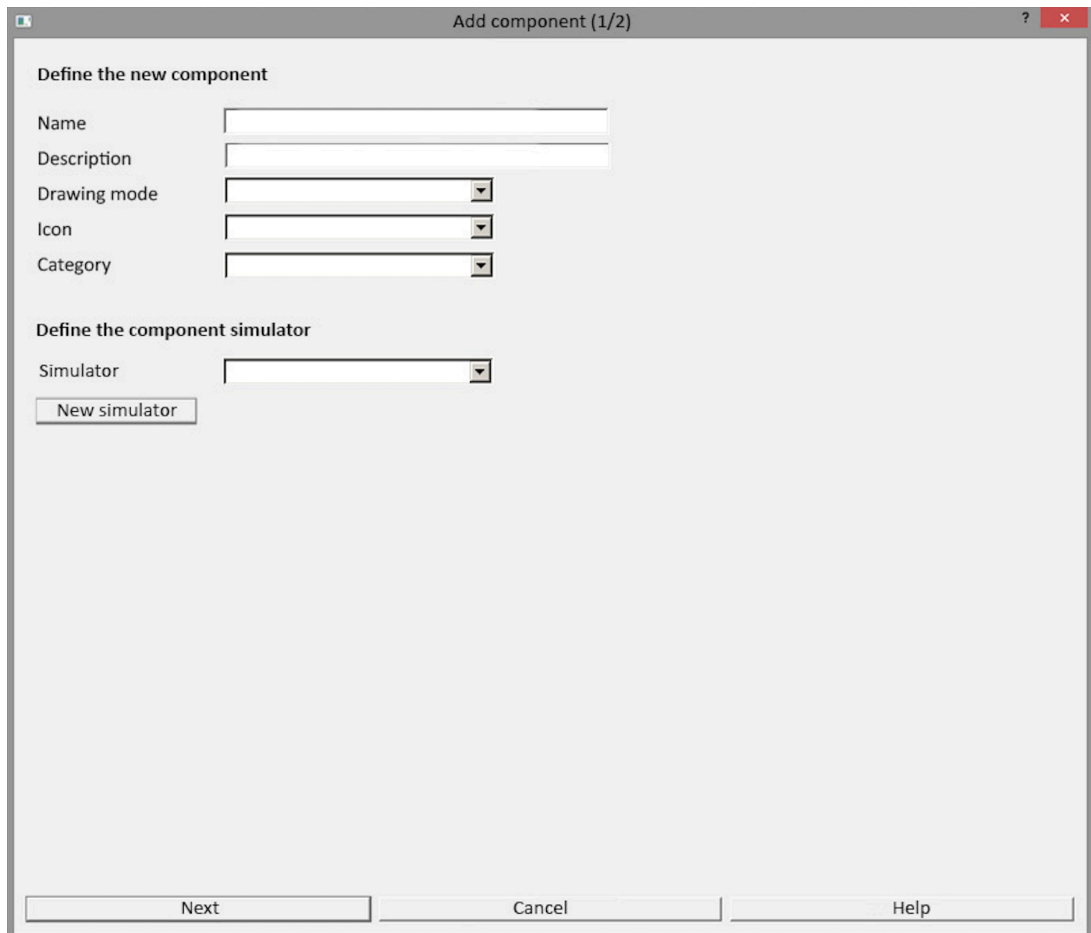


Abbildung 5.7.: Mock-Up der Simulationsansicht

## 2. Mock-Ups Komponentenassistent

In den folgenden Grafiken sind die Mock-Ups für den Komponentenassistenten-

ten zu sehen. Der Komponentenassistent ermöglicht das offene Löschen und Neuanlegen von Simulatoren und Komponenten. Da der Komponentenassistent relativ komplex ist, waren drei Mock-Ups nötig.



The screenshot shows a dialog box titled "Add component (1/2)". It is divided into two main sections:

- Define the new component:** This section contains five input fields:
  - Name: A text input field.
  - Description: A text input field.
  - Drawing mode: A dropdown menu.
  - Icon: A dropdown menu.
  - Category: A dropdown menu.
- Define the component simulator:** This section contains one dropdown menu labeled "Simulator" and a button labeled "New simulator".

At the bottom of the dialog, there are three buttons: "Next", "Cancel", and "Help".

**Abbildung 5.8.:** Mock-Up des Komponentenassistenten

In 5.8 ist die erste Seite zu sehen, in der Grundinformationen der zu erstellen- den Komponente und die Simulatoren auswahl vom Benutzer zu bestimmen sind.

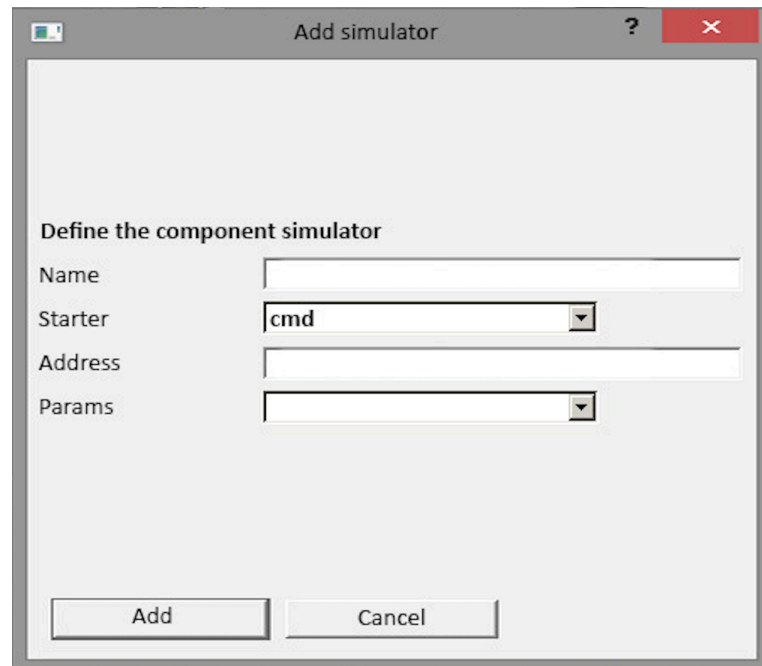


Abbildung 5.9.: Mock-Up zum Hinzufügen eines Simulators

Die Grafik 5.9 zeigt den Dialog, um einen neuen Simulator hinzuzufügen.

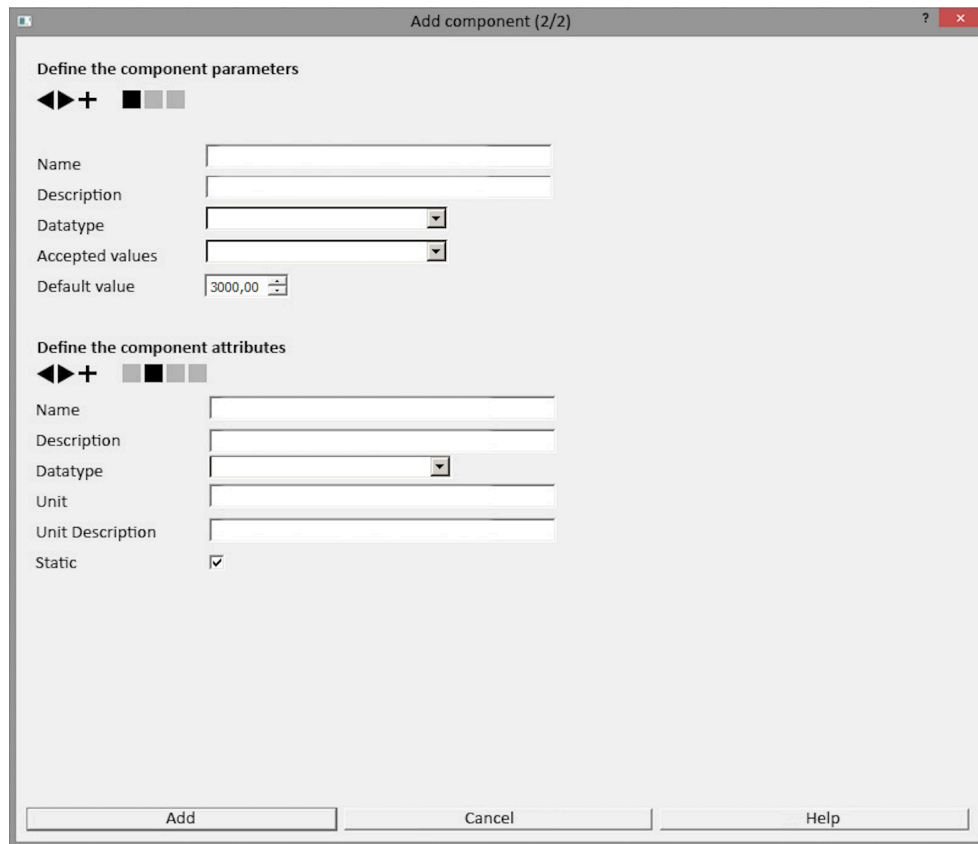


Abbildung 5.10.: Mock-Up zum Hinzufügen von Attributen oder Parametern

Das Mock-Up, zu sehen in der Grafik 5.8, ist etwas umfangreicher als die vorherigen Mock-Ups, da ein generisches Tabulatorenmodell für das Anlegen, Modifizieren und eventuelle Löschen der Attribute und Parameter nötig war.

### 3. Mock-Up EV

Bei der Darstellung des aktuellen Zustands der Komponente Elektrofahrzeug (EV) besteht die Besonderheit, dass Energie aus dem Netz entnommen sowie Energie in das Netz eingespeist werden kann. Auf Basis unserer drei Darstellungsformen (Schatten, Leiste und Transparenz) wurden verschiedene Mock-Ups erstellt und eine Umsetzungsentscheidung getroffen. An dieser Stelle soll die Entscheidung erläutert werden.

#### *Leiste*

Bei der Darstellungsvariante *textitLeiste* wird durch einen Balken, der oberhalb bzw. unterhalb des Komponentensymbols angeordnet ist, visuell dargestellt, mit welcher Menge Energie in das Netz eingespeist oder verbraucht (aus-

gespeist) wird. Bei Komponenten wie Haushalt oder PV-Anlage ist jeweils nur eine Variante möglich, weshalb wir uns generell dafür entschieden haben, die Einspeisung von Energie unterhalb des Icons und den Verbrauch oberhalb des Icons anzuordnen. Der Ladezustand des EV wird durch ein Akkusymbol innerhalb des Icons dargestellt. Hierbei symbolisiert ein Blitz, dass der Akku aufgeladen wird. Zudem ist während des Ladevorgangs der Balken oberhalb des EV-Icons aktiv, da Energie aus dem Netz entnommen (verbraucht) wird. Den aktuellen Stand des Ladevorgangs wird über die Färbung des Akkusymbols dargestellt. Hierbei wird der Akku „grüner“, abhängig von der gespeicherten Energiemenge. Ein geladener Akku wird komplett grün dargestellt und ein vollständig entladener Akku weiß. Wird das EV nicht mehr geladen, sondern speist es Energie in das Netz ein, wird der Ladezustand des Akkus zusätzlich durch einen genauen Prozentwert dargestellt. Bei Einspeisung ist der untere Balken aktiv, um die Höhe der Einspeisung zu visualisieren.

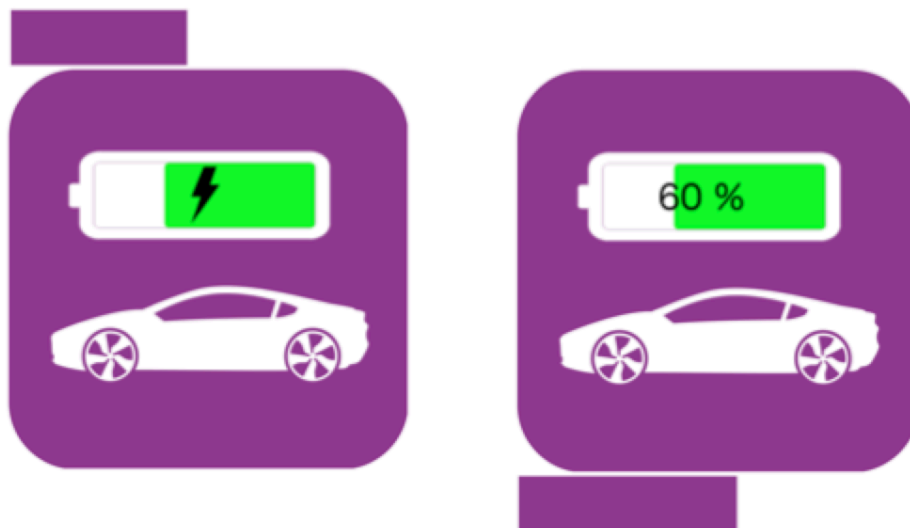


Abbildung 5.11.: Mock-Up Darstellungsvariante *Balken*

### *Schatten*

Bei der Darstellungsform *Schatten* ist die Differenzierung von Ein- bzw. Auspeisung visuell schwer darzustellen, da keine konkrete Trennung wie bei der Balkendarstellungsform möglich ist. Die Schattenstärke stellt hierbei die Höhe der eingespeisten bzw. verbrauchten Energie dar. Eine Option ist, die Schatten jeweils oben oder unten anzuordnen. Wir haben uns allerdings gegen diese Möglichkeit entschieden, da sie im Vergleich mit den anderen Elementen

zu Verwirrung führen könnte .

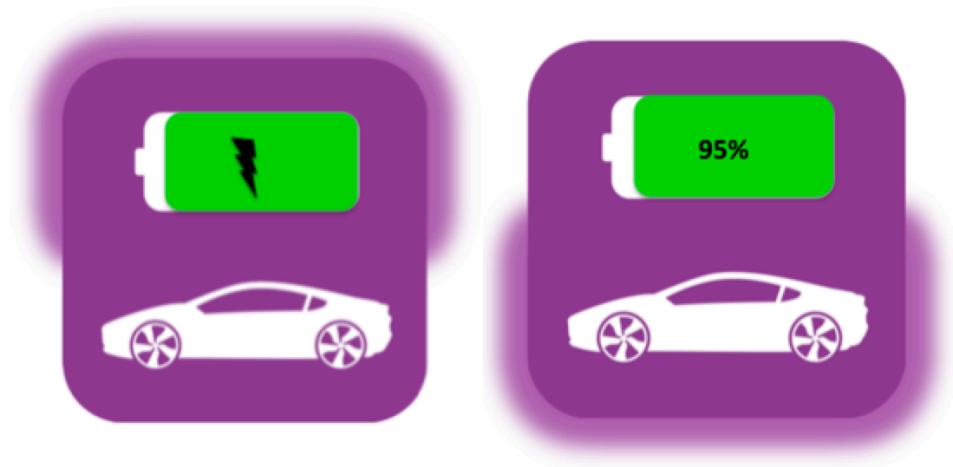


Abbildung 5.12.: Mock-Up Darstellungsvariante *Schatten halbiert*

Wir haben uns deswegen dafür entschieden, bei den Schatten keine visuelle Differenzierung zwischen Ein und Ausspeisung vorzunehmen. Wird der Akku geladen (visualisiert durch den Blitz), stellt der Schatten die Höhe der momentan verbrauchten Energie dar. Bei der Entladung des Akkus (Blitz wechselt zur Prozentanzeige) visualisiert der Schatten die Höhe der aktuell eingespeisten Energie.

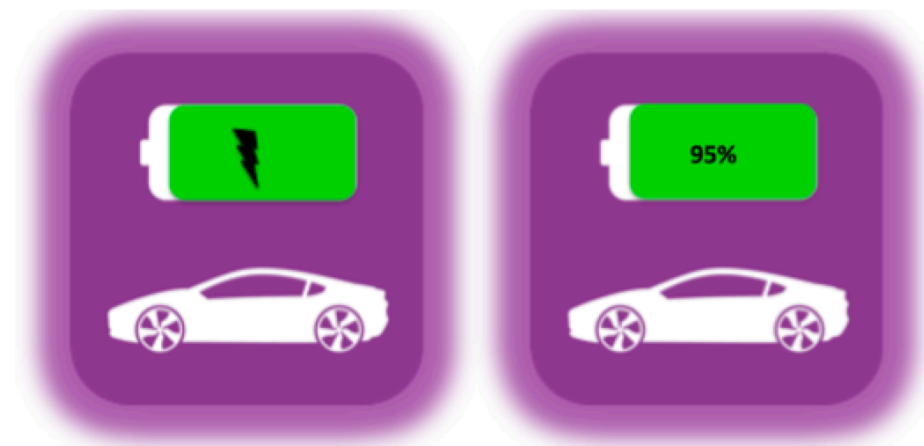


Abbildung 5.13.: Mock-Up Darstellungsvariante *Schatten*

### *Transparenz*

Bei der Darstellungsform *Transparenz* verhält es sich wie bei der Schatten-Darstellungsform. Eine visuelle Trennung von Ein- bzw. Auspeisung lässt sich nicht klar visualisieren und würde unseren konsistenten Ansatz der jeweiligen Darstellungsform je Element widersprechen. Wir haben uns deswegen auch hier gegen eine getrennte Darstellungsform entschieden.



## 6. Implementierung

Im Folgenden wird eine Beschreibung der Paketstruktur des *Maverig* Systems gegeben. Eine gewisse Paketstruktur ist durch die Nutzung des Model-View-Presenter Architekturschemas bereits vorgegeben.

### **Paket *maverig.data***

Dieses Paket beinhaltet statische Klassen, die als Konfigurationsdateien, Internationalisierungsdateien oder als Statusnachrichten dienen. Außerdem sind Grafikdateien für die Anwendung in diesem Paket abgelegt. Auch die Verwaltung der Standard-Pfade für Grafiken oder Internationalisierung, sind in dem Modul *data-Handler* zu finden.

Das Unterpaket *maverig.data.components* beinhaltet die individuellen Komponentenbeschreibungen, die in der Anwendung verwendet werden können. Dazu gehören auch die Beschreibung der jeweiligen Simulatoren mit den benötigten Initialisierungs-Parametern. Von den Beschreibungen benötigte Hilfsklassen, wie beispielsweise der PyPower-Serialisierer und die eigene Implementierung für einen Haus-CSV-Simulator, liegen in dem untergeordneten Paket *maverig.data.components.utils*.

### **Paket *maverig.models***

Dieses Paket enthält die Modellklassen für die Verwaltung und Repräsentation des Anwendungszustandes.

### **Paket *maverig.views***

Dieses Paket enthält die Klassen, die dem Teilbereich der Viewklassen zugeordnet werden. Es existieren, wie bei dem Presenter, Gruppenklassen für die zu nutzenden Komponenten. Die konkrete Darstellung einer Komponente im Szenario wird im Paket *maverig.views.items* definiert. Die Logik für Positionierung und eventueller Kollisionsbehandlung ist im Unterpaket *maverig.views.positioning* zu finden. Wie bereits im Presenter sind auch hier die Viewklassen für die Bedienung der Anwendung zu finden.

### **Paket `maverig.presenter`**

Dieses Paket enthält den Teilbereich der Presenterklassen. Wie zuvor im Entwurf beschrieben, nutzt die Anwendung Presentergruppen, zu denen die einzelnen Energie-Komponenten zugeordnet und im Unterpaket `maverig.presenter.group_presenter` zu finden sind. Die von der Anwendung zur Bedienung genutzten Presenter, wie beispielsweise für die Menübar oder den Eigenschaftsbereich, befinden sich ebenfalls in diesem Paket.

### **Paket `maverig.utils`**

Dieses Paket umfasst universelle Hilfsklassen, die theoretisch auch von anderen Anwendungsdomänen verwendet werden können. Hierzu gehört beispielsweise der ForceAtlas 2-Algorithmus.

### **Paket `maverig.tests`**

Dieses Paket beinhaltet die Testklassen, mit denen die Funktionsfähigkeit und Korrektheit der Anwendung überprüft und garantiert wird. Auf eine genauere Beschreibung wird in den folgenden Unterkapiteln der Übersichtlichkeit halber verzichtet.

Da die Architekturdesignentscheidung zu der Verwendung des Model-View-Presenter Musters führte, orientiert sich die Dokumentation der Implementierung ebenfalls an diesem Konzept. In den folgenden Unterabschnitten wird die wesentliche Implementierung der Anwendung beschrieben und erläutert. Eine ausführliche und detaillierte Code-Dokumentation kann der Entwicklerdokumentation entnommen werden.

## **6.1. Model**

Das Modell der Anwendung ist ein wesentlicher Bestandteil des Funktionsumfangs. Es existieren verschiedene Modellklassen für Komposition, Simulation und Graph-Layout.

### **`maverig.models.model.py`**

Das wesentliche Modell der Anwendung befindet sich in `maverig.models.model.py`. In diesem Modul werden sämtliche Daten, die für die Szenarioabwicklung nötig sind, verwaltet. Dies umfasst beispielsweise alle Elemente, Simulationsstart, Simu-

lationsende, Zwischenablage, ausgewählte Elemente oder die Verwaltung der vier Modi Komponentenmodus, Simulationsmodus sowie Selektionsmodus und Handmodus. Zusätzlich gibt es Validierungsmethoden, die überprüfen, ob bestimmte Elemente im Kompositionsmodus miteinander verbunden werden dürfen oder die das Szenario vor dem Simulationsstart auf Fehlerfreiheit prüfen.

Ein wichtiger Bestandteil des Szenarios ist die Verwaltung von Elementen, die jeweils die aktuellen Daten einer bestimmten Komponente enthalten. Im Modell werden diese Elemente auf Basis von Komponentenbeschreibungen erstellt und bearbeitet, welche in Abschnitt 6.4 noch ausführlich beschrieben werden. Die benötigten Daten der Elemente sind auf das Wesentliche beschränkt und können dadurch auch in gespeicherten Szenarien (.mvrgr-Dateien) übersichtlich dargestellt werden.

```

1  "CSV.House_1": {
2    "elem_id": "CSV.House_1",
3    "mosaik_full_id": "CSV-0.House_1",
4    "sim_model": "CSV.House",
5    "icon": "house2-4.svg",
6    "docking_ports": {
7      "0": {
8        "in": [],
9        "out": [],
10       "pos": [550.0, 440.0]
11     },
12     "1": {
13       "in": [],
14       "out": [{"PyPower.PQBus_6", "0"}],
15       "pos": [495.0, 385.0]
16     }
17   },
18   "params": {
19     "datafile": "maverig/tests/data/household_3_4.small.csv"
20     "P_max": 3354,
21     "num_res": 4,
22     "num_hh": 3,
23     "sim_start": null,
24   },
25   "attrs": {
26     "P": [1080.1, 592.67, 645.99],
27     "P_max": 3354,
28     "num_res": 4,
29     "num_hh": 3
30   }
31 }

```

**Listing 6.1:** Element-Repräsentation eines Hauses im Modell (Auszug aus einer .mvrgr-Szenario-Datei)

In Listing 6.1 ist beispielhaft eine Element-Repräsentation eines Hauses aufgeführt, welches die folgenden Bestandteile umfasst:

- *elem\_id*: Die automatisch generierte ID zur eindeutigen Identifikation und Mapping des Elements.
- *mosaik\_full\_id*: Die zur Simulationslaufzeit in Mosaik vorhandene ID, um die



relevanten eingehenden Daten eindeutig diesem Element zuordnen zu können.

- *sim\_model*: Das Simulationsmodell als Referenz auf die entsprechende Komponentenbeschreibung. Theoretisch wäre hierdurch bei der Weiterentwicklung von Maverig auch ein einfacher Wechsel der zugrunde liegenden Mosaik-Modelle und -Simulatoren möglich, sofern die fehlenden Parameter im Element ergänzt werden und ggfs. das Icon ausgetauscht wird.
- *icon*: Das aktuelle Icon für die Darstellung im Szenario.
- *docking\_ports*: Die Anschlussstellen und deren Positionen sowie eingehende und ausgehende Verbindungen als Referenzen (*elem\_id* und *port*) zu Anschlussstellen anderer Elemente.
- *params*: Die Parameternamen und aktuellen Werte
- *attrs*: Die Attributenamen und während der Simulation gesammelten Werte. Dynamische Werte werden als Liste gespeichert. Attribute, die in derselben Bezeichnung auch als Parameter vorliegen, werden während der Simulation in *maverig.utils.visSimulator.py* automatisch mit den Parameterwerten gefüllt.

Eine Besonderheit der Anwendung ist die Verwendung eines **Event-Konzepts**. Die Verwendung von *Model-View-Presenter* macht es nötig eine Logik zu schaffen, die es ermöglicht, Änderungen an den Daten der Simulation den verschiedenen daran interessierten Komponenten mitzuteilen. Wird beispielsweise der Modus der Komposition geändert, erhält das Modell diese Information und erstellt ein Event, das die daran interessierten Komponenten informiert. Diese Komponenten können beispielsweise die Darstellung von Bedienelementen daraufhin anpassen.

Das Event-Konzept greift dabei auf eine Klassenimplementierung zurück, über welche die am Event interessierten Komponenten an- und abgemeldet sowie auch die Events gefeuert werden. Abbildung 6.1 zeigt die Implementierung des Event-Konzepts auf Klassenebene abstrahiert auf. Die Klasse *Model* greift dabei auf die Klasse *Event* zurück und erstellt für jede Art eines Events eine Instanz dieser Klasse. In der Abbildung sind beispielhaft die Events *selection\_event* für Änderungen der Selektion, *mode\_event* für das Ändern des Modus und *elements\_event* für Änderungen an den Elementen im Modell erfasst. Zusätzlich enthält das Modell die Liste *events\_order*, in der alle Events geordnet nach deren Wichtigkeit enthalten sind. Die Liste wird in den beiden Methoden *update* sowie *update\_all* des Modells dazu genutzt, die Events nach der vorgegebenen Reihenfolge zu feuern. Der Unterschied zwischen beiden Methoden besteht darin, dass die Methode *update* nur geforderte Events feuert, während die Methode *update\_all* sämtliche Events feuert. Ob ein Event gefordert wurde, wird anhand des Flags *demanded* festgestellt, welcher

in der Event-Klasse enthalten ist. Über die Methode *demand* kann ein Event entsprechend gefordert werden. Die Event-Klasse hält in der Liste *\_\_handlers* alle am Event interessierten Komponenten, die beim Feuern des Events zu benachrichtigen sind. Das An- und Abmelden von Komponenten geschieht über die beiden Methoden *handle* und *unhandle* der Event-Klasse. Über die Methode *fire* der Event-Klasse wird das Event mit den entsprechenden Argumenten schließlich gefeuert.

Abbildung 6.1 enthält weiterhin die Implementierung der beiden Presenter-Klassen *ScenarioPanelPresenter* und *ToolbarPresenter*, die aufzeigen, wie das Anmelden an einem Event funktioniert. Um zum Beispiel den *ScenarioPanelPresenter* am Modus-Event (*mode\_event*) anzumelden, ist lediglich die Zeile *self.model.mode\_event += self.on\_mode* erforderlich. Dies ist möglich, weil die Standard-Operatoren „+“ und „-“ in der Event-Klasse mit der Anmelde- beziehungsweise der Abmelde-Methode überschrieben sind. Im Falle dieses Beispiels ruft der *ScenarioPanelPresenter* die Methode *on\_mode* auf, wenn das Modus-Event gefeuert wird. Mit allen anderen Events verhält es sich synonym zum aufgezeigten Beispiel.

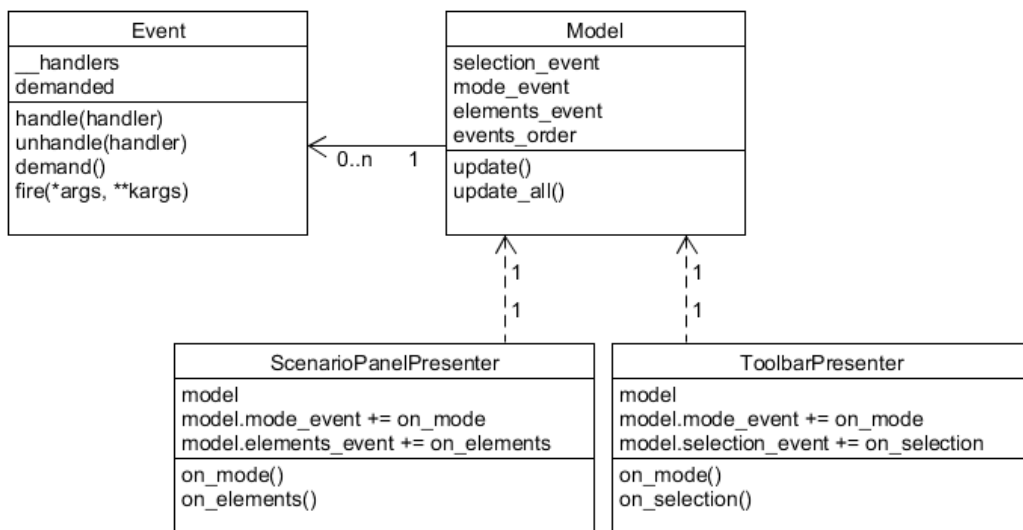


Abbildung 6.1.: Implementierung des Event-Konzepts auf Klassenebene.

### maverig.models.modelSimulation.py

Wenn ein Benutzer ein Szenario erstellt hat und die Simulation startet, wird im Hintergrund ein Simulationsprozess gestartet, um die Anwendung nicht zu blockieren. Dabei stellt das Untermodell *SimulationServer* Methoden zur Modellaktualisierung und Fehlerbehandlung bereit. Diese Methoden sind vom Simulationsprozess aus über ein *ServerProxy*-Objekt aufrufbar. Der Simulationsprozess sendet Funkti-



onspakete mittels der von ZeroMQ<sup>1</sup> bereitgestellten Pair-Kommunikation<sup>2</sup> zum Simulationsserver. Der Simulationsserver fängt die Funktionspakete in regelmäßigen Abständen per Timer ab und ruft die entsprechenden Modellmethoden auf, um Simulationsdaten und Attributwerte zu aktualisieren oder ggfs. Fehlermeldungen anzuzeigen. Für die Attributwerte-Aktualisierung im Modell muss das Mapping der vollen IDs der in Mosaik erzeugten Entitäten auf Maverig-Elemente vorliegen.

Im Simulationsprozess werden generisch alle Mosaik-Modelle und -Simulatoren anhand der Komponentenbeschreibungen und Simulatorbeschreibungen gestartet. Für zu generierende Parameter und Parent-Elemente (für den PyPower-Simulator) gibt es Sonderbehandlungs-Routinen, welche bereits bei den Komponentenbeschreibungen vorliegen. Es werden nur so viele Simulatoren instanziiert, wie auch tatsächlich benötigt sind. Bei mehreren Elementen mit identischen Simulatorbeschreibungen wird derselbe Simulator benutzt.

In der Komposition der Anwendung ist es möglich, Leitungen und Transformatoren auf offline zu setzen. Allerdings ist es anschließend nicht möglich, die Simulation zu starten. Dieser Schutz wurde mit Bedacht eingebaut. Grund dafür ist, dass in Mosaik noch nicht die Möglichkeit besteht, durch die Simulatoren, einen Ausfall einer Leitung oder eines Transformators zu simulieren. Demnach muss zunächst diese Möglichkeit in Mosaik umgesetzt werden, bevor es in Maverig Anwendung finden kann, da Mosaik als Basis dient.

### **maverig.models.modelGraph.py**

Für das konkrete Szenario existiert das Modul *maverig.models.modelGraph.py*. Als wesentliche Bestandteile existieren die Ecken und Kanten des Szenario-Graphen, welche die verschiedenen Komponenten und Verbindungen untereinander repräsentieren. Es ist nicht nur der Aufbau des Graphen wichtig, sondern ebenfalls die Positionen. Dabei verwenden wir eine zweidimensionale Repräsentation der Elemente. Die Daten des Graphen werden außerdem vom ForceAtlas 2-Algorithmus zur Layout-Optimierung genutzt. Die wesentlichen Funktionen dieses Moduls sind das Erstellen des Graphen und die Aktualisierung der Positionen.

## **6.2. View**

Da unsere Anwendung eine Benutzungsoberfläche zur Erstellung von Smart Grid Simulationen ist, erhält die Oberfläche besondere Aufmerksamkeit und entscheidet über die Akzeptanz bei den Benutzern. Die Entscheidung für das Model-View-

---

<sup>1</sup>ZeroMQ Website: <http://zeromq.org>

<sup>2</sup>ZeroMQ PAIR: <http://learning-0mq-with-pyzmq.readthedocs.org/en/latest/pyzmq/patterns/pair.html>



Presenter Konzept erlaubt es uns, auf Wunsch einfach die Oberfläche auszutauschen, da sämtliche Logik und die Daten nicht in der View gespeichert sind. Folgende Bedienungselemente sind durch eine View in unserer Anwendung vertreten:

- Attributebereich
- Konsolenausgabe
- Menübar
- Modus- und Komponentenauswahl
- Fortschrittsleiste
- Eigenschaftensfeld
- Szenariobereich
- Einstellungsfenster
- Statusbar und
- Toolbar.

Zusammen mit unterschiedlichen Dialogen und dem Komponenten-Installationsassistenten befinden sich diese Views im sogenannten Hauptfenster.

Die Abbildung 6.2 stellt das Viewkonzept als Klassendiagramm dar. Im folgenden wird auf die einzelnen Bestandteile des Viewkonzepts genauer eingegangen und weitere Zusammenhänge werden anhand der Abbildung 6.2 erläutert. Die bereits erwähnten Views für die Bedienungselemente werden alle mit der Klasse *MainWindow* verbunden, welche dem Hauptfenster entspricht und somit alle verfügbaren Views im Pfad *maverig/views* beinhaltet. Des Weiteren wird in dieser View das gesamte Layout des Hauptfensters sowie die Anordnung aller anderen Views definiert.

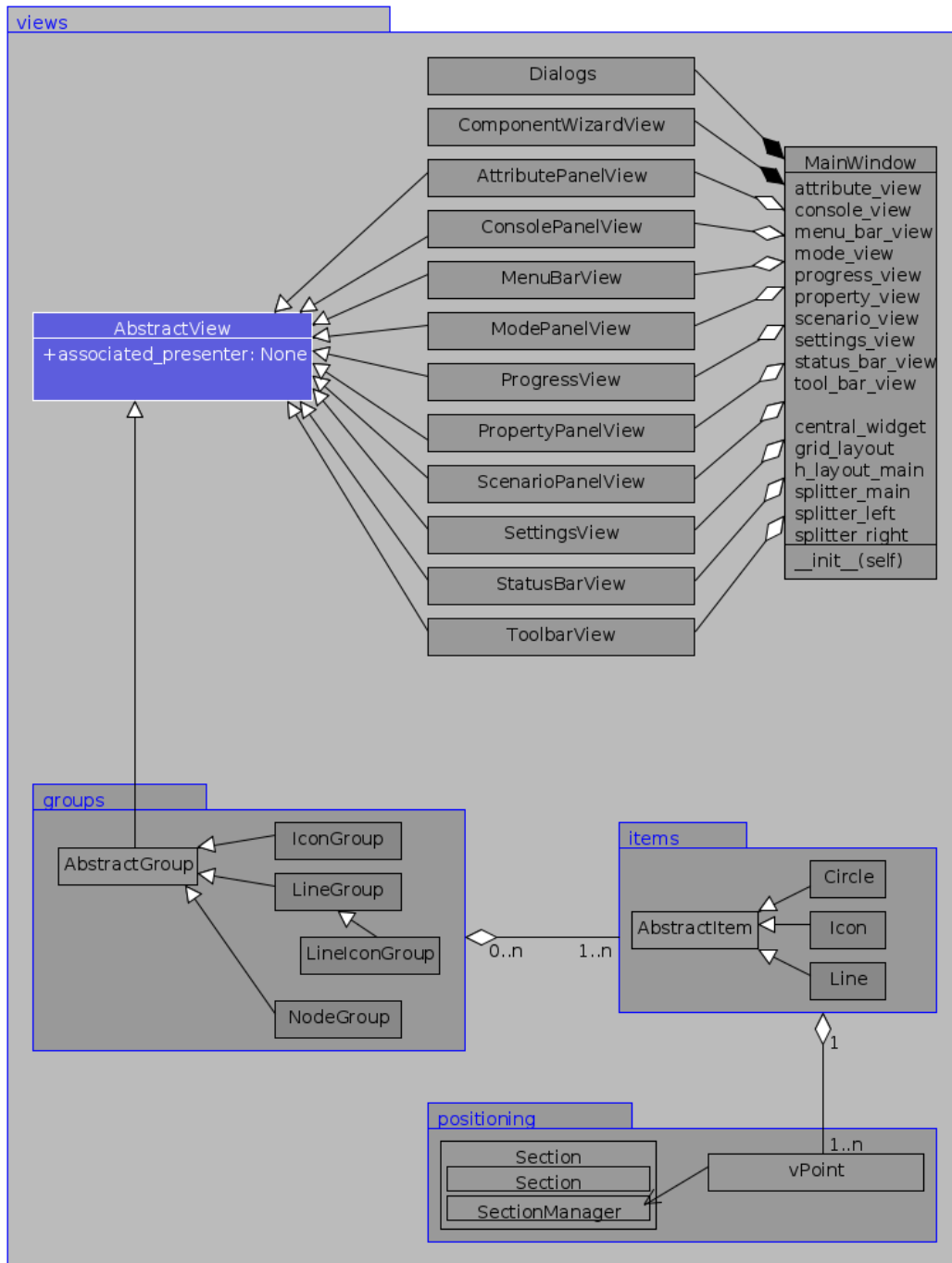


Abbildung 6.2.: Implementierung des View-Konzepts auf Klassenebene.



### **maverig.views.dialogs.py**

In dem Modul *maverig.views.dialogs.py* sind sämtliche in der Anwendung implementierte Dialoge realisiert. Beispielsweise können die Info-Dialoge sowie der Dialog zur Festlegung des Start- und Endzeitpunktes der Simulation hier verändert werden.

### **maverig.views.componentWizardView.py**

In dem Modul *maverig.views.componentWizardView.py* erfolgt die Implementierung des Komponenten-Installationsassistenten zum Hinzufügen von neuen Komponenten in Maverig. Hierbei werden die zu beschreibenden Bestandteile von Simulatoren und Komponenten in einem modalen GUI-Assistenten editierbar gemacht. Die einzelnen Bestandteile werden im Abschnitt 6.4 noch ausführlich erklärt und daher an dieser Stelle nur kurz aufgelistet. Dazu gehören Informationen, wie das entsprechende Mosaik-Simulationsmodell instanziiert wird und welche Darstellungsformen, Parameter und Attribute für die Visualisierung und Modifikation zu Verfügung stehen.

Die Beschreibung einer neuen Komponente umfasst den Simulator, das Modell und das Icon sowie den Darstellungsmodus, die Kategorie und einen Tooltip der neuen Komponente. Die Definition der Komponentenparameter besteht jeweils aus dem Namen, der Beschreibung, dem Datentyp, der erlaubten Werte und des Standardwertes für diese Komponente. Die Komponentenattribute sind entweder statisch oder dynamisch und werden durch den Namen, die Beschreibung sowie die Einheit definiert. In einem zusätzlichen Dialog können neue Simulatoren hinzugefügt und deren Name, Startmethode, Adresse und Parameter mit Standardwerten definiert werden.

Ein- und ausgehende Verbindungsrestriktionen und Trigger vor Parameteränderungen oder Simulationsstart wurden allerdings noch nicht im Assistenten integriert und können dafür aber per Hand in den JSON-Dateien der Komponenten (*maverig.data.components*) und Simulatoren (*maverig.data.components.simulators*) editiert werden.

### **maverig.views**

Wie bereits bei der Beschreibung der Paketstruktur erläutert, existieren für alle Bedienungselemente eigene Views. Diese Views beinhalten die Objekte und Elemente, die in der Maverig-Oberfläche dargestellt werden. Außerdem behandeln sie Benutzereingaben, die den jeweiligen Objekten zugeordnet werden und leiten die Ereignisse an den jeweiligen Presenter weiter. Durch diese strikte Trennung sind die Viewklassen sehr schlank und beinhalten kaum nennenswerte Algorithmen. Wie in

Abbildung 6.2 zu erkennen, erbt jede Viewklasse von *AbstractView*, das allgemein benötigte Eigenschaften und Funktionen bereitstellt.

### **maverig.views.groups**

Zur besseren Strukturierung im Rahmen der Erweiterbarkeit sowie Programm-codereduktion haben wir uns dazu entschieden, Gruppen für verschiedene Komponentenelemente zu realisieren. Es existiert die Klasse *AbstractGroup*, die sämtliche Item-Gruppenklassen implementiert. Dieses Modul enthält Logik und Attribute für die Elemente, die im Szenario genutzt werden. Beispiele hierfür sind das Überprüfen, ob die Maus über einem Element liegt oder das Entfernen eines Elements. Es existieren vier Gruppen für die Szenarioelemente Icon, Linie, Linien-Icon und Knoten. In Abbildung 6.2 handelt es sich dabei um die *IconGroup*, die *LineGroup* und die *NodeGroup*. Die Klasse *LineIconGroup* erbt hierbei von der Klasse *LineGroup*.

### **maverig.views.items**

Wie zuvor beschrieben, existieren für Elemente verschiedene Gruppen. Den Gruppen werden die eigentlichen Items *Kreis*, *Icon* und *Linie* zugeordnet. In den Klassen dieses Moduls wird das Aussehen der Elemente und die Eigenschaften, wie Verbindungen zu anderen Punkten und Position, implementiert. Daraus ergibt sich eine Verbindung der einzelnen Items zu den verschiedenen Gruppen, wie in Abbildung 6.2 dargestellt. Den Item-Klassen *Circle*, *Icon* und *Line* werden alle allgemein benötigten Eigenschaften und Funktionen durch die Klasse *AbstractItem* bereitgestellt. Die verschiedenen Gruppen können sich aus *einem* oder *n* Items zusammensetzen, wohingegen ein Item zu *keiner* oder *n* Gruppen gehören kann.

### **maverig.views.positioning**

Für die Positionierung von Komponenten auf die Zeichenfläche wurde ein spezielles Konzept entwickelt. Für jedes Element werden virtuelle Punkte erzeugt, die unter dem entsprechenden Element liegen. Durch dieses Konzept ist die Positionierung der Elemente geregelt. Demnach verändern sich die Positionen der virtuellen Punkte, wenn das Element verschoben wird. Es können auch mehrere Elemente miteinander verknüpft (Docking) werden. Das heißt, wenn ein Element verschoben wird und diese z.B durch eine Leitung mit einem anderen Element verbunden ist, verschiebt sich das andere Element ebenfalls entsprechend. Des Weiteren werden die Positionen der Elemente auch aktualisiert, wenn sie sich am das Raster ausrichten. Bei Positionsänderungen werden allgemein mehrere Zustandsebenen, auch Sections genannt, durchlaufen. Dabei werden Positionsänderungen der nächsthöheren Ebene (Section) erst durchgeführt, wenn alle Positionsänderungen der unteren Ebe-



nen abgeschlossen sind.

Die virtuellen Punkte (vPoints) passen sich somit nach einer Umpositionierung durch die Maus als erstes gegenseitig an. Danach werden in der Item-Section die einzelnen Items der Elemente auf ihre entsprechenden vPoints gesetzt. In der Presenter-Section reagieren dann die GroupPresenter und lösen nicht mehr geltende Verbindungen (Dockings) und Ausrichtungen am Raster auf.

Für diese Funktionalität existieren die Module *maverig.views.positioning.vPoint.py* und *maverig.views.positioning.section.py*, welche die einzelnen unterschiedlichen Abschnitte der Verschiebung bzw. Aktualisierung der Position der vPoints behandeln. Die Abbildung 6.2 zeigt, dass jedes Item *einen* oder *n* vPoints beinhalten kann, während ein vPoint immer zu einem bestimmten Item gehört. Des Weiteren nutzt die Klasse *vPoint* den *SectionManager* des Moduls *maverig.views.positioning.section.py* zur Iteration der verschiedenen als Sections bezeichneten Zustandsebenen.

### 6.3. Presenter

Die Presenter stellen die wesentliche Logik der Anwendung bereit und dienen als Schnittstelle zwischen den Views (siehe 6.2) und dem Modell (siehe 6.1).

#### **maverig.presenter**

Zu jeder vorhandenen View existiert ein zugeordneter Presenter. Jeder Presenter empfängt Ereignisse, welche über die View oder über das Modell ausgelöst werden und führt die entsprechende Logik aus. Alle Presenter implementieren eine Basis-Klasse, um allgemein benötigte Attribute und Funktionen nutzen zu können. Bei der Instanziierung erhalten alle Presenter einen Verweis auf die Klasse *PresenterManager* (siehe folgenden Abschnitt), das Modell und die Konfigurationsdatei (siehe 6.4). Das Koppeln der Presenter und Views passiert im Einstiegspunkt der Anwendung.

#### **maverig.presenter.presenterManager.py**

Die Klasse *PresenterManager* enthält Verweise auf alle existierenden Presenter. Dadurch ist es möglich, dass eine Kommunikation zwischen den Presentern stattfindet. Dies dient dazu, Logik an zuständige Presenter delegieren zu können. So delegiert beispielsweise der Presenter für die Menüleiste viele der verfügbaren Funktionen an zuständige Presenter weiter. Wenn z.B. die Konsole über die Menüleiste versteckt wird, leitet der Presenter für die Menüleiste diese Aufgabe an den Presenter der Konsole weiter. Zum einen sollen dadurch die Codezeilen auf alle Presenter gleichmäßig verteilt sowie die Views betreffende Logik in den jeweils zuständigen Presentern ausgeführt werden. Zum anderen soll die Anzahl an Events überschaubar bleiben.



### **maverig.presenter.group\_presenter**

Dieses Paket enthält Presenter für die Komponenten. Die Funktionsweise dieser Presenter verhält sich synonym zu den bisher beschriebenen Presentern. Das Instanzieren erfolgt beim Erstellen eines Elements im Szenario.

### **maverig.presenter.utils.forceEngine.py**

Die Graphenoptimierung des Layouts wird innerhalb des *ScenarioPanelPresenter* über die Hilfsklasse *ForceEngine* vorgenommen. Diese Hilfsklasse verwaltet und initialisiert den ForceAtlas 2-Algorithmus (*maverig.utils.forceatlas2.py*). Zur Initialisierung gehören z.B. die Anzahl der Iterationen und der in *maverig.models.modelGraph.py* generierte *NetworkX*-Graph inklusive der Positionsattribute. Die Kräfteberechnung und die darauffolgende Positionierung der virtuellen Punkte wird auf mehrere Iterationen verteilt, welche mithilfe eines Timers ausgelöst werden. Dadurch wird die Anwendung nicht blockiert und es können auch während des Neuansordnens Elemente manuell verschoben werden, um so ein besseres Layout zu erhalten.

## **6.4. Data**

Für die Darstellung der Anwendung und auf der Modell-Ebene werden verschiedene Daten aus diesem Paket eingebunden. Dazu gehören die Icons, die Komponenten- und Simulatorbeschreibungen und Konfigurationsdaten wie Standard-Texte und -Pfade.

### **maverig.data.config.py**

Die *maverig.data.config.py* stellt Methoden bereit, mit denen persistente Daten im JSON-Format ausgelesen und gespeichert werden können. Außerdem sind an dieser Stelle einige Standardtexte und Konstanten gespeichert, welche in der Anwendung für verschiedene Zwecke Verwendung finden. Diese Datei kann zudem ausgeführt werden, um neue Spracheinträge aus den Komponentenbeschreibungen in die Übersetzungsdateien (.po-Dateien) hinein zu generieren. Dies ist insbesondere dann ein nützliches Tool, wenn neue Komponenten hinzugefügt werden, deren Inhalte in verschiedene Sprachen übersetzt werden sollen.

### **maverig.data.dataHandler.py**

Die *maverig.data.dataHandler.py* stellt Methoden bereit, um Pfadangaben zu relativen Pfaden zu verkürzen (*get\_relpath*) oder zu vollen Pfaden abhängig vom Maverig-Ordner zu erweitern (*get\_normpath*). Die relativen Standardpfade für die Komponenten, Simulatoren, Komponentenicons, Anwendungsicons, Sprachpfade und Konfigurationsdateien können dabei betriebssystemunabhängig ausgehend vom Maverig-Ordner definiert werden, z.B. *maverig/data/languages/* als Suchpfad für die Sprachdateien.

## Persistenz der Komponenten- und Simulatorbeschreibungen

Das Paket *maverig.data.components* enthält die zur Komposition und Simulation benötigten Komponentenbeschreibungen als JSON-Dateien, welche entweder direkt per Hand oder im Komponentenwizard erstellt werden können. Es gibt bereits einige vorhandene Komponentenbeschreibungen (Referenzbus, Transformator, Bus, Kabel, Photovoltaikanlage, Windkraftanlage, Blockheizkraftwerk, Haus, Elektroauto), die in dem Paket als JSON-Dateien vorliegen.

```

1 {
2   "creation_time": "2014-01-03T00:01:00",
3
4   "sim_model": "CSV.House",
5
6   "type": ["Component", "CSV", "House"],
7   "category": "Consumers",
8   "tooltip": "House",
9
10  "icon": "house.svg",
11  "drawing_mode": "icon",
12
13  "docking_ports": {
14    "0": {},
15    "1": {"out": ["PQBus"]}
16  },
17
18  "on_sim_init": null,
19  "on_set_param": "maverig.data.components.utils.simInit:on_set_param_house",
20
21  "published_params": ["P_max", "datafile"],
22  "published_attrs": ["P_max", "P", "num_hh", "num_res"],
23
24  "params": {
25    "sim_start": {},
26    "P_max": {
27      "caption": "Maximum Power",
28      "datatype": "float",
29      "default_value": 1118
30    },
31    "num_hh": {
32      "default_value": 1
33    },
34    "num_res": {
35      "default_value": 2
36    },

```

```

37     "datafile": {
38         "caption": "CSV data file",
39         "datatype": "file (*.csv)",
40         "default_value": "maverig/tests/data/household_1_2.small.csv"
41     }
42 },
43
44 "attrs": {
45     "P": {
46         "caption": "Active Power",
47         "unit": "W",
48         "static": false
49     },
50     "P_max": {
51         "caption": "Maximum Power",
52         "unit": "W",
53         "static": true
54     },
55     "num_hh": {
56         "caption": "Households",
57         "unit": "",
58         "static": true
59     },
60     "num_res": {
61         "caption": "Residents",
62         "unit": "",
63         "static": true
64     }
65 }
66 }

```

**Listing 6.2:** Komponentenbeschreibung für Häuser

Als Beispiel wird in Listing 6.2 eine Komponente für Häuser beschrieben, die einen Simulator für CSV-Datenreihen benutzt. Die Beschreibung einer Komponente umfasst folgende Bestandteile:

- *creation\_time*: Das Erstelldatum für die Sortierung der Komponenten im Moduspanel.
- *sim\_model*: Auswahl des Mosaik-Simulators und -Modells getrennt durch einen Punkt. Modelle können durch Namenserverweiterung nach einem Bindestrich mehrfach beschrieben werden (z.B. CSV.WECS-Offshore).
- *type*: Eine hierarchische Liste an Typbezeichnungen, welche als referenzierbare Klassifikationen für die erlaubten Komponentenverbindungen (Dockings) dienen.
- *category*: Kategoriebezeichnung für eine übersichtliche Gliederung nach Kategorien im Moduspanel.
- *tooltip*: Eine Kurze Komponentenbeschreibung für Tooltips im Moduspanel.
- *icon*: Der Dateiname des Komponenten-Icons unter *maverig/data/components/icons*. Erlaubt sind SVG-, PNG- und JPG-Dateien.



- *drawing\_mode*: Darstellungsmodus der Komponente als Linie, Icon, Knoten oder Linie mit zentriertem Icon ('line', 'icon', 'node' oder 'line-icon-line').
- *docking\_ports*: Anschlussstellen und jeweils erlaubte Verbindungen an eingehenden ('in') und ausgehenden ('out') Typen (siehe *type*-Bezeichnung oben).
- *on\_sim\_init*: optionale Methodenreferenz der Form *Moduladresse:Methodenname* mit Parametern *model* und *element*, die im Simulationsprozess noch vor dem Starten des Elements ausgeführt wird. Hierdurch werden bspw. in *maverig.data.components.utils.simInit:on\_sim\_init\_powertransmitter* die Parameter 'fbus' und 'tbus' von Transformatoren und Branches vor der Simulation mit den Element-IDs der verbundenen Busse initialisiert.
- *on\_set\_param*: Optionale Methodenreferenz der Form *Moduladresse:Methodenname* mit Parametern *model*, *element* und *param\_name*, die vor jedem Setzen eines Parameters in Maverig ausgeführt wird. So werden z.B. bei jeder Dateiänderung durch Setzen des Parameters 'datafile' in Listing 6.2 in *maverig.data.components.utils.simInit:on\_set\_param\_house* die Parameter für die Anzahl der Haushalte und Residenten sowie das entsprechende Haus-Icon und ggfs. die Leistungsobergrenze entsprechend der Standardlastprofile aktualisiert.
- *published\_params*: Parameter, die im PropertyPanel editierbar sein sollen.
- *published\_attrs*: Attribute, die im AttributePanel während der Simulation angezeigt werden sollen.
- *params*: Die Mosaik-konformen Parameternamen mit den jeweiligen Bezeichnungen, Datentypen, erlaubten Werten und Standardwerten. Relative vom Maverig-Ordner ausgehende Dateipfade werden im Simulationsprozess in *modelSimulation.py* automatisch vervollständigt. Ebenso wird auch der spezielle Parameter *sim\_start* dort erst mit der in Maverig eingestellten Simulationsstartzeit initialisiert.
- *attrs*: Die Mosaik-konformen Attributenamen mit den jeweiligen Bezeichnungen, Einheiten und ob diese statisch oder dynamisch sind. Zusätzlich können ein- ('in') oder ausgehende ('out') Attributverbindungen je Attribut definiert werden (z.B. 'P': { ..., 'out': ['P\_in'] }), welche bei aneinander gedockten Elementen als Attributverbindungen in Mosaik übertragen werden.

```

1 {
2     "name": "CSV",
3     "starter": "python",
4     "address": "maverig.utils.maverig_csv:CSV",
5     "params": {

```



```

6     "sim_start": null,
7     "datafile": null
8   },
9   "on_sim_init_parents": null
10 }

```

**Listing 6.3:** CSV-Simulator

In *maverig.data.components.simulators* liegen die von den Komponenten referenzierten Simulatorbeschreibungen (PyPower und CSV). In diesen wird beschrieben, wie und mit welchen Parametern der entsprechende Mosaik-Simulator gestartet werden soll. In Listing 6.3 wird der CSV-Simulator *maverig.utils.maverig\_csv* beschrieben, welcher statische und dynamische Attributwerte aus einer CSV-Datei ausliest.

- *name*: Der Name entspricht dem referenzierten Simulatornamen aus den Komponenten, welcher nach einem Bindestrich erweitert werden kann, um verschiedene Simulatoreinstellungen zu ermöglichen (z.B. PyPower-Excel).
- *starter und address* ist die Referenz zum Mosaik-Simulator entsprechend der Simulatorkonfiguration in Mosaik<sup>3</sup>
- *params*: Die Mosaik-konformen Parameternamen mit den jeweiligen Standardwerten. Parameter, deren Namen bereits in der Komponentenbeschreibung vorhanden sind, werden vor der Simulation mit den Komponentenparameterwerten gefüllt.
- *on\_sim\_init\_parents*: Optionale Methodenreferenz der Form *Moduladresse:Methodenname* mit Parametern *model* und *element*, die in der Simulation vor dem Starten von Non-Public-Modellen für einen Simulator ausgeführt werden. Auf diese Weise kann z.B. in *maverig.data.components.utils.simInit:on\_sim\_init\_parents\_pypower* eine JSON-Datei für den PyPower-Simulator generiert werden, um damit ein übergeordnetes Grid-Modell zu starten.

Die Komponenten- und Simulatorbeschreibungen dienen dem Modell (*maverig.models.model.py*) in Abschnitt 6.1 als Grundlage zum Erstellen und Bearbeiten von Elementen, welche daraufhin bei der Simulation in *maverig.models.modelSimulation.py* als Mosaik-Simulatoren und -Modelle instanziiert werden.

### Persistenz von Anwendungseinstellungen

Das Speichern von Anwendungseinstellungen erfolgt in einer Konfigurationsdatei, welche unter *maverig/data/configs/cfg.json* zu finden ist. Es handelt sich bei dem Format demnach um JSON. Damit die Konfigurationsdatei angelegt wird, ist das

<sup>3</sup>Simulatorkonfiguration in Mosaik: <http://mosaik.readthedocs.org/en/latest/simmanager.html>



erstmalige Ausführen der Anwendung notwendig. Nach dem erstmaligen Ausführen erstellt die Anwendung die Konfigurationsdatei mit den Standardeinstellungen, die unter *maverig/data/settings/defaultSettings.py* hinterlegt sind. Dies beugt gleichzeitig Fehlern vor, wenn die Konfigurationsdatei bspw. vom Nutzer gelöscht wird.

Die Anwendungseinstellungen sind in der Konfigurationsdatei in verschiedene Kategorien gruppiert. Zum Zeitpunkt der Anfertigung dieser Dokumentation sind folgende Kategorien vorhanden:

**ui\_state** Die unter dieser Kategorie geführten Einstellungen betreffen den Zustand der grafischen Oberfläche, wie Fenstergröße oder Sichtbarkeit der einzelnen Panels.

**simulation\_settings** Einstellungen, welche die Simulation betreffen, sind unter dieser Kategorie geführt. Dabei handelt es sich um Einstellungen wie Einfärbungen der Szenario-Elemente während der Simulation. Diese Einstellungen können im Optionsmenü unter dem Reiter *Simulation* verändert werden.

**general\_settings** Unter dieser Kategorie sind die allgemeinen Einstellungen, wie bspw. die Sprache, aufgeführt. Diese können im Optionsmenü unter dem Reiter *Allgemein* verändert werden.

**mode\_panel\_settings** Diese Kategorie enthält Einstellungen, die das ModusPanel betreffen. Darunter fallen Einstellungen wie das Ein- oder Ausblenden unsichtbarer Komponenten.

Der Aufbau der Konfigurationsdatei mit den Kategorien und Einstellungen gestaltet sich wie folgt, hier stark verkürzt:

```

1 { "ui_state": {
2     "main_window_geometry": "AdnQyw..",
3     "main_window_state": "AAAA/w..",
4     "is_property_panel_visible": true,
5     ...
6 },
7 "simulation_settings": {
8     "is_heat_value_effect_for_grids_enabled": true,
9     "heat_value_effect_grids": "Color",
10    ...
11 },
12 "general_settings": {
13     "language": "de_DE",
14     ...
15 },
16 "mode_panel_settings": {
17     "invisible_components": ["CSV.CHP"],
18     "show_invisible_components": false
19 }}

```

**Listing 6.4:** Verkürzter Aufbau der Konfigurationsdatei

Für den Zugriff auf die Konfigurationsdatei bietet die Datei *maverig/data/config.py* die Methoden *read\_config* zum Auslesen und *write\_config* zum Beschreiben der Konfigurationsdatei. Die Datei enthält zudem die Klasse *ConfigKeys*, welche die Schlüssel zum Zugriff auf die Kategorien sowie Einstellungen der Konfigurationsdatei hält. Beim Anwendungsstart wird die Konfigurationsdatei damit an benötigten Stellen über die Methode *read\_config* ausgelesen und die Einstellungen über die Schlüssel in der Klasse *ConfigKeys* ausgelesen. Beim Ändern von Einstellungen wird zunächst über die Schlüssel auf die entsprechende Einstellung zugegriffen, der neue Wert gesetzt und die Konfigurationsdatei anschließend über die Methode *write\_config* mit der geänderten Konfiguration überschrieben. Beim Ändern von Einstellungen wird zusätzlich das Event *settings\_event* im Model (siehe 6.1 für Erläuterung des Event-Systems) gefeuert, damit die anderen Instanzen die geänderte Konfigurationsdatei neu auslesen und die Änderungen ebenfalls übernehmen.

Zur Verdeutlichung des zuvor beschriebenen Umgangs mit der Konfigurationsdatei soll ein Beispiel dienen. Für das Beispiel soll das Auslesen und Manipulieren der Einstellung *is\_property\_panel\_visible* unter der Kategorie *ui\_state* (siehe Listing 6.4) betrachtet werden. Diese Einstellung gibt an, ob das Eigenschaftenspanel für Szenario-Elemente sichtbar ist. Das Eigenschaftenspanel ist nur im Kompositionsmodus sichtbar, falls der Nutzer es nicht ausblendet. Im Simulationsmodus ist das Panel dagegen nie sichtbar. Beim Anwendungsstart befindet sich die Anwendung im Kompositionsmodus, d.h. hierbei muss aus der Konfigurationsdatei gelesen werden, ob das Panel sichtbar ist. Dafür ist die zugehörige Presenter-Klasse *PropertyPanelPresenter* unter *maverig/presenter/propertyPanelPresenter.py* zuständig. Die Presenter-Klasse bekommt die Konfigurationsdatei beim Anwendungsstart von der Klasse *PresenterManager* unter *maverig/presenter/presenterManager.py* überreicht, welche an die Variable *self.cfg* gebunden wird. Der Presenter reagiert auf das Event *program\_mode\_event* aus dem Modell, welches beim Anwendungsstart bzw. beim Ändern des Anwendungsmodus gefeuert wird. Die Methode, mit welcher der Presenter auf das Event reagiert, behandelt die Sichtbarkeit des Eigenschaftenspanels beim Wechsel des Anwendungsmodus und ist wie folgt implementiert:

```

1 def on_program_mode(self):
2     if self.model.program_mode == ProgramMode.composition: # Anwendung ist im
3         Kompositionsmodus
4         self.view.setHidden(not self.cfg[ConfigKeys.UI_STATE][ConfigKeys.
5             IS_PROPERTY_PANEL_VISIBLE]) # Sichtbarkeit wird auf den Wert aus der
6             Konfigurationsdatei gesetzt
7     elif self.model.program_mode == ProgramMode.simulation: # Anwendung ist im
8         Simulationsmodus
9         self.view.hide()

```

Listing 6.5: Reaktion des *PropertyPanelPresenter* auf das Event *program\_mode\_event*

Wenn die Anwendung also in den Kompositionsmodus wechselt, wird die Sichtbarkeit des Eigenschaftenspanels aus der Konfigurationsdatei mit

`self.cfg[ConfigKeys.UI_STATE][ConfigKeys.IS_PROPERTY_PANEL_VISIBLE]` durch die entsprechenden Schlüssel ausgelesen und auf diesen Wert gesetzt. Wechselt die Anwendung in den Simulationsmodus, ist das Auslesen nicht notwendig, da das Panel in diesem Modus nie sichtbar sein soll. Ändert der Nutzer die Sichtbarkeit des Panels im Kompositionsmodus, reagiert der Presenter mit der Methode `on_change_visibility_triggered` auf die Änderung. Die Implementierung der Methode gestaltet sich wie folgt:

```

1 def on_change_visibility_triggered(self):
2     self.view.setHidden(not self.view.isHidden()) # setzt Sichtbarkeit des Panels
3     self.presenter_manager.menu_bar_presenter.view.action_trigger_property_panel.
4         setChecked(
5             not self.view.isHidden()) # aktualisiert die Sichtbarkeit in der Menüleiste
6     self.cfg[ConfigKeys.UI_STATE][ConfigKeys.IS_PROPERTY_PANEL_VISIBLE] = not self.view
7         .isHidden() # setzt den neuen Wert in der gehaltenen Konfiguration
8     config.write_config(self.cfg) # überschreibt Konfigurationsdatei mit gehaltener
9         Konfiguration
10    self.model.settings_event.demand() # fordert das Event settings_event
11    self.model.update() # feuert das Event settings_event, um Änderungen bekannt zu
12        machen

```

**Listing 6.6:** Reaktion des *PropertyPanelPresenter* auf das Ändern der Sichtbarkeit des Eigenschaftenspanels

Die ersten beiden Zeilen der Methode behandeln das Setzen der Sichtbarkeit, was für den Umgang mit der Konfigurationsdatei unwesentlich ist. Über die Zeile 5 in Listing 6.6 wird der geänderte Wert für die Sichtbarkeit in der vom Presenter gehaltenen Konfiguration geändert. Danach erfolgt das Überschreiben der Konfigurationsdatei mit der bereits erwähnten Methode `write_config`. Damit alle anderen Instanzen die geänderte Konfigurationsdatei neu einlesen, wird abschließend über die letzten beiden Zeilen das Event `settings_event` gefeuert. Das Auslesen von Einstellungen über die Schlüssel, das Überschreiben der Konfigurationsdatei und das Bekanntmachen der Änderungen verhält sich an allen anderen Stellen im Anwendungscode synonym zum aufgeführten Beispiel.

## 6.5. Util

Das Util-Paket enthält eine Ansammlung diverser Klassen, die keine zentralen Kernelemente der Software darstellen, dieser aber bei der Ausführung diverser Aufgaben behilflich sind. Es greifen Elemente der gesamten MVP-Architektur vereinzelt auf Klassen dieses Pakets zu. Durch ihre logische Abkapselung von der restlichen Struktur sind diese Klassen theoretisch auch von äußeren Anwendungen je für sich nutzbar.

### **maverig.utils.colorTools.py**

Das Modul *maverig.utils.colorTools.py* ist für die Visualisierung von Farbkomponenten verantwortlich. Bereits bekannt ist, dass die Visualisierung von energetischen Auslastungen der Elemente während der Simulation in einem Koordinatensystem stattfindet. Mithilfe dieser Methodiken können Farben der darzustellenden Elemente im Szenario extrahiert und in je leicht abgewandelter Form den Linien des Koordinatensystems zugeordnet werden. Dies ermöglicht eine Mehrfachdarstellung mehrerer gleichartiger Elemente in der Attributeansicht, ohne dabei die Fähigkeit zu verlieren, einzelne Elemente unterscheiden zu können.

### **maverig.utils.event.py**

Die in dem Modul enthaltene Klasse *Event* ist elementarer Bestandteil des MVP-Konzeptes, in dem es darum geht, Änderungen im Model, der View oder im Presenter den jeweils nicht betroffenen Komponenten kenntlich zu machen. Solche Modifikationen lösen das Feuern eines Event aus. Wenn dieses ausgeführt wird, sorgt dies für eine Aktualisierung beteiligter Klassen mit den neuen Daten. Der genaue Ablauf dieses Konzeptes ist im Abschnitt der Modell-Implementierung beschrieben.

### **maverig.utils.flowlayout.py**

*maverig.utils.flowlayout.py* ist ein eigens implementiertes Modul, das die Funktionalitäten von PyQt 4.x's *FlowLayout* abdecken soll. Diese Klasse an sich ist nicht in PySide enthalten. Da der Bedarf an Layouts, die sich in der Höhe und Breite beschränken lassen und in der anderen Richtung mit Inhalt expandieren können, allerdings gegeben ist, wurde dieses Layout übernommen.

### **maverig.utils.forceatlas2.py**

Generell wird ForceAtlas 2 als Optimierungsalgorithmus benutzt, um bestehende Graphen mit einer Menge von Knoten und Kanten im Layout zu optimieren. Hierauf wurden im Kapitel der eingesetzten Software, sowie in einer zugrundeliegenden Seminararbeit [Kla14a] eingegangen.

Das referenzierende Modul hierauf, *maverig.utils.forceatlas2.py*, besteht aus den reinen Initialisierungsmethoden für die Liste der Graphenpunkte, für die einzelnen Gewichte der Graphen, insofern diese vorgegeben sind. Als Folge dieser Daten ist die Länge der Kanten zusätzlich lieferbar. Diese beiden Methoden werden von der übergeordneten Methode *init\_data(g, nodelist, dim, edge\_weight)* genutzt, um den Gesamtgraphen mit seinen einzelnen Knoten und Kanten, sowie dessen Gewichten aufzubauen.

Die untergeordnete Klasse *SpeedModel* dient als Berechnungsgrundlage, die die globale und lokale Anpassungsgeschwindigkeit des Algorithmus bestimmt. Es speichert als kleines Modell alle geschwindigkeitsrelevanten Parameter ab. Als lokale Geschwindigkeit gilt die Geschwindigkeit, die sich aus der vorgegebenen globalen Anpassungsgeschwindigkeit und zusätzlich der Positionierung eines Knotens im Graphen gemessen an den anderen errechnet. Diese ist gesondert methodisch abrufbar.

Wichtigste Unterklasse ist *ForceAtlas 2*. Diese Klasse nutzt soeben vorgestellte Methodiken sowie das Speed-Modell und weitere Einstellungsparameter wie die Skalierung, Mindestabstände zum Vermeidung von Knotenüberlappungen, Dimensionen, oder erweiterte Berechnungsverfahren wie den LinLog-Algorithmus, welcher für eine verengte Darstellung von Knoten in Clustern sorgt. Nach Grundinitialisierung aller Parameter sorgt die Methode *do\_layout* für die Neuberechnungen aller Knoten pro Iteration und aktualisiert den Gesamtgraphen.

### **maverig.utils.logger.py**

*maverig.utils.logger.py* ist ein Modul zum Logging, die verwendet wird, um besondere Vorkommnisse im System als Log-Event festzuhalten. Für unseren Fall wurde die Klasse *StreamToLogger* benutzt, um entstehende Logging-Texte nicht in Text-Dateien zu speichern, sondern sie direkt sichtbar für den Nutzer in der Konsole auszugeben. Dieses Vorgehen finden bspw. im Modell für die Simulationsvorgänge Anwendung.

### **maverig.utils.maverig\_csv.py**

*maverig.utils.maverig\_csv.py* ist eine Kopie des Mosaik-CSV-Simulators, nur dass diese um statische und zusätzlich dynamische Datenunterstützung erweitert wurde. Bereits in der Initialisierung einer solchen Datei wird zwischen statischen und dynamischen Attributen unterschieden, da erstere feste Attributwerte, letztere allerdings Attributwerte enthalten, die jeweilig einem Datum zugeordnet werden können. Es ist also möglich, Simulationsschritt für Simulationsschritt gemessen an einer Zeitfolge darstellen zu können. Für sich ändernde Werte kann dieses Modul daher den Inhalt für das je nächste Datum liefern, sodass eine dynamische Datendarstellung ermöglicht wird. Dies geschieht solange, bis das Ende der CSV-Datei erreicht wurde.

### **maverig.utils.numTools.py**

*maverig.utils.numTools.py* ist ein Hilfsmodul, welche es erlaubt, einen String-Wert als seinen gewünschten Datentyp wie *float* oder *int* zurückzugeben. Bei der Übergabe von normalen Zahlen kann diese Klasse diese mit passenden Präfixen versehen oder ab einer gewissen Länge von Ziffern als Kurzschreibweise zurückgeben.

### **maverig.utils.processServer.py**

*maverig.utils.processServer.py* unterstützt mit *Server* und *ProcessServer* Konzepte des Multiprocessing. Die *Server*-Klasse kann eine Vielzahl an Funktionen registrieren, um diese einem Subprozess über einen Proxy (*ServerProxy*) bereitzustellen. Bei der Inter-Prozess-Kommunikation sendet der Subprozess in *MethodProxy* über einen ZeroMQ-Socket Funktionspakete (*Call\_Pack*) an die *Server*-Klasse *ProcessServer*, welche die Funktionspakete mit dem Methodennamen und Parameterwerten per Timer in regelmäßigen Abständen abgreift und als Methodenaufruf ausführt. Diese Konzepte finden im Simulationsprozess und Simulationsserver des Simulationsmodells Anwendung, welche im Modell-Kapitel Abschnitt 6.1 beschrieben wurden.

### **maverig.utils.scenarioErrors.py**

*maverig.utils.scenarioErrors.py* enthält eine Ansammlung mehrerer Unterklassen, die im Szenario auftretende Fehler dem Nutzer visualisieren sollen. Sie enthalten einen Titel, einen an die Konsole angepassten Text, einen normalen Text und einen Infotext. Im Rahmen der Szenariovalidierung oder im Falle von fehlerhaften Verhalten während der Simulation werden die Klassen im Rahmen eines jeweiligen Exceptionhandlings gefeuert. Dies führt zum Aufruf einer Exceptionhandling-Methode, die mithilfe der Parameter eine Konsolenausgabe oder eine andere Visualisierung auslöst.

### **maverig.utils.tableWidgets.py**

*maverig.utils.tableWidgets* setzt sich aus *CellLineEdit* und *AutoRowTableWidget* zusammen. Letztere Unterklasse ist eine klassische Tabelle der QtGui-Palette, welche allerdings die letzte Reihe immer frei lässt und sich automatisch im Index hochzählt, wenn eine neue Zellenkomponente in diese eingefügt wurde. *CellLineEdit* wird bei jedem neuen Element in dieser Tabelle registriert und kann auf gedrückte Tastaturbefehle und Textänderungen reagieren. Dies stellt sicher, dass bei einer völligen Löschung der letzten Reihe diese auch wieder im Index zurückgezählt wird, sodass dieses Widget eine dynamische Anpassung ermöglicht.

### **maverig.utils.visSimulator.py**

*maverig.utils.visSimulator.py* stellt nach ähnlichem Schema wie *maverig\_csv.py* verbesserte Darstellungsmethoden für dynamische Werte zur Verfügung. Die Speicherstruktur gibt in diesem Fall allerdings vor, dass in einer Vielzahl von Modellen



eine Untermenge an Topologien gespeichert werden, die für sich wiederum Parameter und Attribute enthalten. Der Simulator wird mit einem Startdatum, einem Proxy, einer Liste von Elementen, der ID und der Schrittgröße der Simulation initialisiert. Mit jedem Schritt kann wie zuvor eine übergebene Zeit referenziert und auf den aktuell darzustellenden Elemente bezüglich ihrer Werte aktualisiert werden. Geänderte Daten können dann über den Simulationsproxy aktualisiert werden. Da dieser, wie bereits angesprochen, nach dem Warteschlangenprinzip arbeitet, werden die Daten in der Reihenfolge ihres Eintreffens verarbeitet, also im Regelfall nach Abfolge der Zeitstempel.

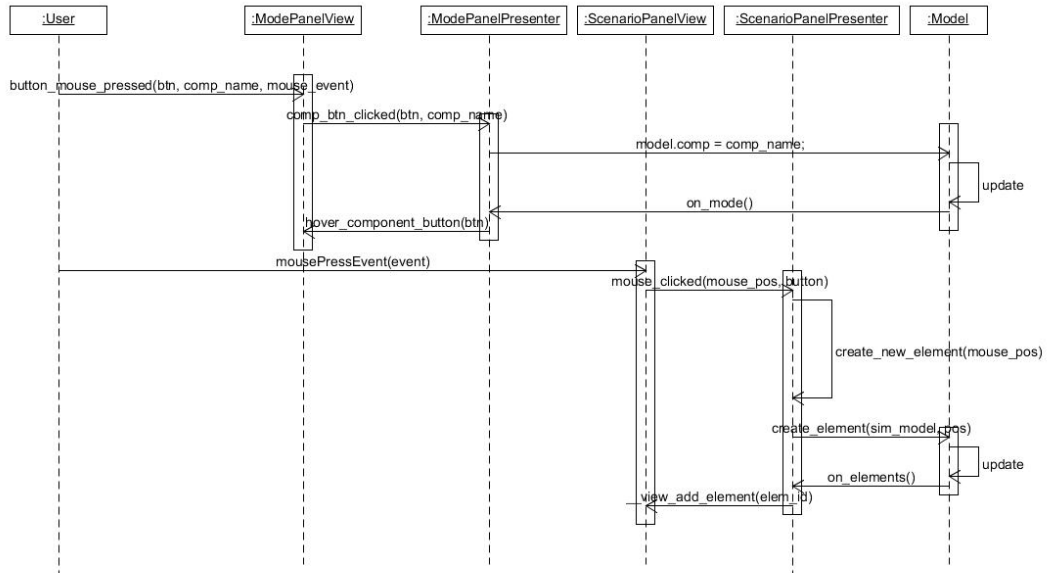
## 6.6. Anwendungsfall

In diesem Beispiel handelt es sich um das Zusammenspiel von dem Model-View-Presenter (MVP) Architekturkonzept. Das Design von MVP unterscheidet sich von MVC (Model-View-Controller) dadurch, dass das Model die View nicht kennt und deshalb nicht direkt mit ihr interagieren kann. Um die Implementierung zu verdeutlichen, veranschaulicht der folgende Anwendungsfall das Erzeugen einer Komponente durch einen Benutzer (z.B. ein Grid).

Nachdem man auf eine Komponente in der View (*ModePanelView*) geklickt hat, nimmt die View diese Aktion wahr und schickt ein Event zu dem jeweils zugeordneten Presenter (*ModePanelPresenter*). Der Presenter berücksichtigt immer, in welchem Modus sich die Anwendung gerade befindet und aktualisiert diese im Model. Das Model ändert also den vorherigen in den nun ausgewählten Modus: Falls man auf einen Komponentenbutton klickt und bereits der Komponentenmodus aktiviert ist, wird der Selektionsmodus aktiviert und dies entsprechend in der Benutzeroberfläche dargestellt. Befand sich die Anwendung nach dem Klick auf einen Komponentenbutton nicht bereits im Kompositionsmodus, wechselt das Model den Modus ebenfalls und gibt die Information über den Presenter an die View weiter. Der geklickte Komponentenbutton wird hervorgehoben. Wenn der Benutzer also eine Komponente ausgewählt hat und sich die Anwendung im Kompositionsmodus befindet, kann der Benutzer in das Szenario (*ScenarioPanelView*) klicken und somit die entsprechende Komponente erzeugen. Dies geschieht in der Implementierung folgendermaßen: Nachdem der Benutzer auf die *ScenarioPanelView* klickt, wird ein Mouse-Event erzeugt und zu dem *ScenarioPanelPresenter* geschickt. Dieser überprüft, ob man die linke oder rechte Maustaste gedrückt hat und in welchem Modus sich die Anwendung aktuell befindet. Wenn der Benutzer die linke Maustaste geklickt hat, schickt der *ScenarioPanelPresenter* eine Information zu dem Model. In dem Model wird die neue Komponente erzeugt, zudem geschieht eine Validierung der Eingabe und es erfolgt die Rückgabe der Element-ID von dem Modell zum Sce-



*narioPanelPresenter*. Anschließend aktualisiert der *ScenarioPanelPresenter* die *ScenarioPanelView*. Das bedeutet, die neue Komponente wird angezeigt.





## 7. Tests und Performance

### 7.1. Testdokumentation

Das Testen ist ein wichtiger Bestandteil, um ein System auf Funktionsfähigkeit und Robustheit zu prüfen. Deshalb ist es unablässig, verschiedene Tests durchzuführen. Während der Sprints *Dokumentation und Tests* wurden die umgesetzten Komponenten getestet und Fehler direkt behoben. Fehler sind hierbei Abweichungen der tatsächlichen Form eines Qualitätsmerkmals von der Soll-Form, Inkonsistenzen zwischen der Spezifikation und der Implementierung und jedes strukturelle Merkmal des Programmtextes, das zu einem fehlerhaften Verhalten im Programm führt. Wie bereits im Kapitel 2.2.2 erwähnt, sollen die Testarten Unittest, Integrationstest, Regressionstest, Systemtest und Sprachtest durchgeführt werden. Diese werden im Folgenden näher beschrieben.

Unittests werden zur Prüfung der verschiedenen Klassen und ihrer Methoden genutzt. Diese bieten die Basis zur Ermittlung von Fehlern im Rahmen der Implementierung. Im Projekt wurden für alle Presenter und die Klassen *model.py*, *modelSimulation.py*, *vPoint.py* und *event.py* Unittests erstellt.

Die Methode *test\_get\_u\_heat\_value* aus der Klasse *test\_model.py* prüft beispielsweise, welche Spannungsabweichung in Prozent eine Komponenteninstanz zu einem spezifischen Zeitpunkt hat. Hierzu wird ein Beispielszenario geladen und ein Simulationszeitpunkt ausgewählt. Die Assert-Methode vergleicht im Anschluss, ob die Werte des PQ-Knoten und der Leitung mit den dort angegebenen Werten übereinstimmen.

```

1 def test_get_u_heat_value (self):
2     """Returns the u_heat_value from an element on a specific timestamp."""
3     self.model.scenario = config.read_json(dataHandler.get_normpath('maverig/tests/data
4         /demo_sim.mvrg'))
5
6     # add simultion data indicies
7     self.model.sim_timestamps = {self.model.sim_start + timedelta(seconds = self.model.
8         sim_step_size*i) for i in range(12)}
9     self.model.sim_index = 0
10
11     assert self.model.get_u_heat_value('PyPower.PQBus_5') == - 0.00373482425657185
12     assert self.model.get_u_heat_value('PyPower.Branch_1') == - 0.0015318751959563226

```

Listing 7.1: Beispieltestfall für die Methode *test\_get\_u\_heat\_value*



Insgesamt wurden (Stand 03.03.2015) 171 Unittests implementiert und somit für die relevanten Pakete eine Testabdeckung von 78 % erreicht. Dies ist für eine Desktopanwendung ein beachtliches Ergebnis, da diese häufig mit vielen View-Elementen verknüpft sind und diese nicht direkt mit Hilfe von Unittests geprüft werden können.

Um alle Tests zu durchlaufen, kann dies über die Kommandozeile im Maverig-Projekt anhand des Befehls `py.test` durchgeführt werden. Um einen Bericht der Code-Abdeckung in den jeweiligen Klassen zu erhalten, muss in die Kommandozeile `py.test --cov=maverig` eingegeben werden. Die Abbildung zeigt eine Coverage-Übersicht der durchlaufenen Codeklassen.

Coverage report: 78%



<i>Module</i>	<i>statements</i>	<i>missing</i>	<i>excluded</i>	<i>coverage</i>
maverig\EntryPoint	85	85	0	0%
maverig\__init__	0	0	0	100%
maverig\data\__init__	0	0	0	100%
maverig\data\components\utils\__init__	0	0	0	100%
maverig\data\components\utils\pyPowerSerializer	27	2	0	93%
maverig\data\components\utils\simInit	41	3	0	93%
maverig\data\config	127	29	0	77%
maverig\data\dataHandler	53	19	0	64%
maverig\data\settings\__init__	0	0	0	100%
maverig\data\settings\abstractSettings	22	6	0	73%
maverig\data\settings\defaultSettings	28	0	0	100%
maverig\data\settings\heatValueEffect	6	0	0	100%
maverig\data\settings\settings	45	0	0	100%
maverig\demo	132	132	0	0%
maverig\models\__init__	0	0	0	100%
maverig\models\model	667	62	0	91%
maverig\models\modelGraph	25	0	0	100%
maverig\models\modelSimulation	194	19	0	90%
maverig\presenter\__init__	0	0	0	100%
maverig\presenter\abstractPresenter	10	0	0	100%
maverig\presenter\attributePanelPresenter	79	2	0	97%
maverig\presenter\componentWizardPresenter	67	48	0	28%
maverig\presenter\consolePanelPresenter	36	0	0	100%
maverig\presenter\group_presenter\__init__	0	0	0	100%
maverig\presenter\group_presenter\abstractGroupPresenter	351	80	0	77%
maverig\presenter\group_presenter\iconGroupPresenter	7	0	0	100%
maverig\presenter\group_presenter\lineGroupPresenter	17	0	0	100%
maverig\presenter\group_presenter\lineIconGroupPresenter	2	0	0	100%
maverig\presenter\group_presenter\nodeGroupPresenter	6	0	0	100%
maverig\presenter\menuBarPresenter	339	95	0	72%
maverig\presenter\modePanelPresenter	160	29	0	82%
maverig\presenter\presenterManager	13	0	0	100%
maverig\presenter\progressPresenter	102	4	0	96%
maverig\presenter\propertyPanelPresenter	125	7	0	94%
maverig\presenter\scenarioPanelPresenter	275	38	0	86%
maverig\presenter\settingsPresenter	51	25	0	51%
maverig\presenter\statusBarPresenter	62	0	0	100%
maverig\presenter\toolbarPresenter	101	15	0	85%

maverig\presenter\utils\__init__	0	0	0	100%
maverig\presenter\utils\forceEngine	39	15	0	62%
maverig\tests\__init__	0	0	0	100%
maverig\tests\test_attributePanelPresenter	99	2	0	98%
maverig\tests\test_consolePresenter	85	0	0	100%
maverig\tests\test_event	23	0	0	100%
maverig\tests\test_groupPresenter	397	1	0	99%
maverig\tests\test_menuBarPresenter	260	0	0	100%
maverig\tests\test_modePanelPresenter	168	7	0	96%
maverig\tests\test_model	355	18	0	95%
maverig\tests\test_modelSimulation	23	0	0	100%
maverig\tests\test_progressPresenter	152	0	0	100%
maverig\tests\test_propertyPanelPresenter	149	0	0	100%
maverig\tests\test_scenarioPanelPresenter	259	2	0	99%
maverig\tests\test_settingsPresenter	85	4	0	95%
maverig\tests\test_statusBarPresenter	119	4	0	97%
maverig\tests\test_toolbarPresenter	93	0	0	100%
maverig\tests\test_vPoint	30	0	0	100%
maverig\utils\__init__	0	0	0	100%
maverig\utils\colorTools	129	56	0	57%
maverig\utils\event	41	6	0	85%
maverig\utils\flowlayout	61	1	0	98%
maverig\utils\forceatlas2	111	47	0	58%
maverig\utils\logger	20	20	0	0%
maverig\utils\maverig_csv	87	10	0	89%
maverig\utils\numTools	34	11	0	68%
maverig\utils\processServer	55	0	0	100%
maverig\utils\scenarioErrors	178	88	0	51%
maverig\utils\tableWidgets	36	27	0	25%
maverig\utils\visSimulator	50	4	0	92%
maverig\views\__init__	0	0	0	100%
maverig\views\abstractView	3	0	0	100%
maverig\views\attributePanelView	278	30	0	89%
maverig\views\componentWizardView	457	407	0	11%
maverig\views\consolePanelView	54	0	0	100%
maverig\views\dialogs	242	222	0	8%
maverig\views\groups\__init__	0	0	0	100%
maverig\views\groups\abstractGroup	76	12	0	84%
maverig\views\groups\iconGroup	22	0	0	100%
maverig\views\groups\lineGroup	15	0	0	100%
maverig\views\groups\lineIconGroup	25	0	0	100%
maverig\views\groups\nodeGroup	6	0	0	100%

maverig\views\items\__init__	0	0	0	100%
maverig\views\items\abstractItem	152	21	0	86%
maverig\views\items\circle	40	2	0	95%
maverig\views\items\icon	39	2	0	95%
maverig\views\items\line	59	7	0	88%
maverig\views\mainWindow	80	80	0	0%
maverig\views\menuBarView	228	0	0	100%
maverig\views\modePanelView	186	40	0	78%
maverig\views\positioning\__init__	0	0	0	100%
maverig\views\positioning\section	38	1	0	97%
maverig\views\positioning\vPoint	86	1	0	99%
maverig\views\progressView	105	0	0	100%
maverig\views\propertyPanelView	216	24	0	89%
maverig\views\scenarioPanelView	119	69	0	42%
maverig\views\settingsView	81	70	0	14%
maverig\views\statusBarView	32	0	0	100%
maverig\views\toolbarView	83	0	0	100%
<b>Total</b>	<b>9115</b>	<b>2011</b>	<b>0</b>	<b>78%</b>

coverage.py v3.7.1

Abbildung 7.1.: Testabdeckung

Somit lässt sich aber nur sicherstellen, dass die Komponenten in sich geschlossen funktionieren. Der Integrationstest ermöglicht Fehler zu identifizieren, die aufgrund des Zusammenwirkens verschiedener Softwaremodule, in diesem Fall Maverig und Mosaik, auftreten können. Bei dem Zusammenwirken verschiedener Komponenten muss auf eine fehlerfreie Interaktion dieser unterschiedlichen Komponenten geachtet werden. Im Unittest wurden bereits Methoden getestet, die mit Mosaik in Verbindung treten, wie der Beispieltest, der oben aufgeführt wurde. Somit kann der Integrationstest auch auf dieser Ebene bereits erfolgen. Neben dem Integrationstest wurden der Systemtest und der Sprachtest mithilfe der Usability-Studien durchgeführt. Eine genauere Dokumentation ist im folgenden Kapitel nachzulesen. Der Systemtest überprüft das Gesamtsystem unter möglichst realistischen Bedingungen. In der Studie wurde darauf geachtet, dass die Teilnehmer dem Endverbraucher entsprechen, die mit dieser Software regelmäßig arbeiten sollen.

In dem Sprachtest geht es um die Überprüfung einer sinngemäßen Bedeutung der unterschiedlich angebotenen Sprachen, sowie die passende Benutzung der Fachbegriffe. Auch dieser Test wurde durch die Teilnehmer sichergestellt und abschließend dokumentiert.

Abschließend wird noch der Regressionstest erwähnt, der die Wiederholung von Testfällen vorsieht. Der Unittest kann wiederholt durchgeführt werden, um sicherzustellen, dass Änderungen die Funktionalität nicht eingeschränkt haben. Damit kann das Regressionstesten auch auf dieser Ebene erfolgen.

## 7.2. Usability-Tests

Um zu überprüfen, inwiefern die Software den Anforderungen der Gebrauchstauglichkeit gemäß ISO 9241-11 entspricht, haben wir uns dazu entschieden zwei Usability-Studien durchzuführen. Dabei ist die Gebrauchstauglichkeit als Empfindung eines spezifischen Nutzers in einem speziellen Nutzungskontext zur Bearbeitung einer festgelegten Aufgabe definiert. Die entscheidenden Parameter sind Effektivität, Effizienz und Zufriedenheit der Nutzer.

Der allgemeine Nutzungskontext entspricht dem Anwendungsgebiet der entwickelten Software. Daraus lässt sich entsprechend die Zielgruppe ableiten, in der sich Personen befinden, die mit der Benutzung derartiger Software bereits vertraut sind. Es kann eine ausreichende elektrotechnische Vorkenntnis vorausgesetzt werden, welche die Benutzer dazu befähigt, valide Szenarien abbilden zu können.

### 7.2.1. Usability Studie I

An der ersten Studie am 10. September 2014 haben vier Personen aus der Zielgruppe teilgenommen. Es ging darum anhand verschiedener Aufgaben die Grundfunktionalitäten zu überprüfen und die von uns darüber hinaus zu diesem Zeitpunkt implementierten Features auf ihre Nützlichkeit zu untersuchen. Zusätzlich zu den Protokollen, die während der Studie geführt wurden, haben wir die Kommentare der Teilnehmer mithilfe eines Diktiergeräts zur späteren Nachbearbeitung aufgezeichnet. Im Anschluss daran wurden die Teilnehmer gebeten einen Fragebogen auszufüllen, auf dem Aussagen in Form einer Likert-Skala bewertet werden sollten. Zu den Aufgaben gehörten das Erstellen eines kleinen validen Szenarios, bei dem Operationen wie Kopieren, Einfügen, Speichern und Laden eines Szenarios sowie die Individualisierung der Nutzeroberfläche gestellt wurden. Die zu bewertenden Aussagen auf den Fragebögen bezogen sich auf die Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlerrobustheit und Individualisierbarkeit der Software.

Die Auswertung der Studie hat neben dem Aufzeigen von Bugs auch Verbesserung der Funktionalitäten sowie Anregungen zur Implementierung neuer Funktionen hervorgebracht und Verbesserungen bezüglich des Designs offen gelegt, die im Folgenden aufgelistet werden.

- Bugs
  - Fehler beim automatischen Ausrichten am Raster.
  - Das Benutzen der ZoomFit-Funktion liefert nicht immer das gleiche Ergebnis.
  - Die Zeichenfläche erweitert sich zu schnell, wenn eine Element an den Rand geführt wird.
  - Beim Laden eines Szenarios gehen Verbindungen teilweise verloren.
  - Sehr langsame Reaktion des PropertyPanels auf Eingaben.
  - Simulationsendzeitpunkt kann zeitlich vor dem Startzeitpunkt gewählt werden.
- Änderungen an Funktionen
  - Das Zoomen soll zusätzlich über den Kombination „Strg + Mausrad“ möglich sein.
  - Nur Mousrad: Scrollen.
  - Strg + Mousrad: Horizontales Scrollen.

- ZoomFit soll so ausgeführt werden, dass die Scrollbars in der Zeichenfläche verschwinden
- Bevor ein neues Szenario geladen wird, soll eine Speicherabfrage geschehen.
- Neue Funktionen
  - Komponenten sollen per Drag& Drop in die Zeichenfläche eingefügt werden können.
  - Hinzufügen von Buttons für das Rückgängig machen und Wiederherstellen in die Toolbar.
  - Vermeiden, dass ungültige Verbindungen gezeichnet werden können.
  - Die Größe der verschiedenen Panels der Software soll individualisierbar sein.
  - Hinzufügen eines neuen Verbindungstyps zwischen PQBus und Konsumenten bzw. Produzenten.
  - Vorschlagen eines Standardpfades für das Speichern von Szenarien.
- Design
  - Das Design der Scrollbars an das Betriebssystem anpassen.
  - Bei Komponenten mit einem Icon auf der Linie, soll die Markierung bei einer Selektion auf das Icon verlegt werden.
  - Vergrößern der ausgewählten Komponente im Modus-Panel.
  - Eindeutigere Kennzeichnung des Selektions- bzw. Hand-Modus.
  - Beschriftung bei der Einstellung des Simulationsstart- und Endzeitpunkts fehlen.
  - Hinzufügen einer Visualisierung des Offline-Modus für Transformatoren

Die aufgelisteten Ergebnisse wurden in der an die Studie anschließenden Sitzung besprochen und in Aufgabenpakete zur Bearbeitung unterteilt. Alle aufgeführten Vorschläge wurden umgesetzt.

Bei der Interpretation der Ergebnisse des Fragebogens wurde deutlich, dass der Grad der Individualisierbarkeit der Benutzeroberfläche zu gering ausfällt und auch die Hilfestellungen seitens der Software nicht ausreichend ist und an verschiedenen Stellen deutlicher hervorgehoben werden sollte.

Der generelle Funktionsumfang sei jedoch zu dem Zeitpunkt der Studie in Ordnung und gut durch das Feedback der Software an den Nutzer unterstützt. Die, sich



an Standards orientierten Shortcuts, seien sinnvoll und nützlich sowie das Beheben von Fehlern und Rückgängig machen fehlerhafter Eingaben einfach. An vielen Stellen werden fehlerhafte Eingaben bereits bei der Eingabe überprüft bzw. nicht zugelassen.

### 7.2.2. Usability-Studie II

An der zweiten Studie, die am 21. Januar 2015 durchgeführt wurde, nahmen sieben Personen teil. Zu den Teilnehmern gehörten neben denen aus dem Bereich der Energieinformationssysteme auch Elektrotechniker, da in dieser Studie neben der Usability auch die Validität der dargestellten Inhalte von Bedeutung war. Die Teilnehmer wurden wiederum gebeten Aufgaben zu bearbeiten, wobei der Fokus dieser Studie mehr auf den bis dahin neu implementierten Teil der Simulation lag. Im Anschluss an die Bearbeitung wurde den Teilnehmern der System-Usability-Scale Fragebogen<sup>1</sup> (SUS-Fragebogen) vorgelegt, anhand dessen ein Usability-Score errechnet werden kann, der Aufschluss über die Gebrauchstauglichkeit eines Softwaresystems gibt.

Die Aufgaben bestanden unter anderem darin, ein valides Szenario zu erstellen und dieses simulieren zu lassen. Während der Simulation sollte sich der Teilnehmer bestimmte Parameter einzelner Elemente anzeigen lassen, sowie festgelegte Fehler provozieren, um deren Behandlung innerhalb der Software zu testen.

Die folgende Auflistung unterteilt sich in die gefundenen Bugs und Fehler im System sowie Features, die geändert bzw. hinzugefügt werden sollten.

- Bugs und Fehler
  - Transformer und RefBus werden nach dem automatischen Layouten zu nah aneinander angeordnet
  - Fehlende Einheit beim Einstellen des Simulationsintervalls.
  - Stufeneinstellung der Simulationsgeschwindigkeit ist zu uneindeutig.
  - Abfangen des Eintragens unzulässiger Simulationsintervalle.
  - Eingabe von Werten im PropertyPanel per Hand teilweise nicht möglich.
  - Zu schwache Hervorhebung von fehlerverursachenden Elementen.
  - Unklare Unterscheidung zwischen Mosaik- und Maverig-Fehlern.
  - Fehlende Achsenbeschriftung in den Graphen der Simulation.
  - Fehlender Verschiebung des AttributePanels auf den gerade aufgeklappten Graphen.
  - Scrollen im AttributePanel nicht bzw. nur teilweise möglich.

<sup>1</sup><https://blog.seibert-media.net/blog/2011/04/11/usablility-analysen-system-usability-scale-sus/>

- Farben der Graphen bei Auswahl mehrerer gleicher Elementarten nicht unterscheidbar genug.
- Legendeneinträge in den Graphen überlappen sich.
- Anzeigefehler bei den verschiedenen Einheiten im AttributePanel.
- Fehlendes Feedback, wenn bei der Auswahl unterschiedlicher Elementarten keine gemeinsamen Attribute im AttributePanel angezeigt werden.
- Bezeichnung „Nominal Power“ bei den Haushalten in der Form nicht korrekt.
- Angaben mit sechs Dezimalstellen zu viel.
- Unterscheidung zwischen den beiden Fortschrittsbalken der Simulation unklar.
- In Tooltips angezeigte Daten teilweise fehlerhaft, speziell bei der Anzeige des aktuellen Datums auf dem Fortschrittsbalken der Simulation.
- Aktualisierung des Start- und Endzeitpunktes der Simulation gelegentlich fehlerhaft.
- Fehlerhafte Anzeige der Simulationsgeschwindigkeit in der Statusbar.
- Beschriftung der Heatvalue-Einstellungen nicht ausreichend.
- Heatvalue-Effekte der Schatten wurden teilweise als zu schwach empfunden.
- Darstellung der Heatvalue-Effekte der Prosumenten bislang nur durch Balken.
- Kurzes Auftauchen und Schließen eines Fensters bei der Auswahl eines Elements während der Simulation.
- Inkonsistenz der Übersetzung.
  - \* Heatvalue-Einfärbung
  - \* Sprache
- Fehlendes Inhaltsverzeichnis im Benutzerhandbuch.
- Gelegentlicher Abbruch der Simulation bei 90-95% Simulationsfortschritt.
- Fehler beim Wechseln der Komponenten- und Handmodi
- Mindestlänge der Transformerkomponente zu gering.

- Wenn nach dem Start der Software ein neues Szenario erstellt wurde, fand eine unnötige Speicherabfrage statt und danach konnten keine Komponenten mehr in das Szenario eingefügt werden.
  - Abfangen fehlerhafter Längenangaben von Leitungen.
  - Fehlerbehandlung der an einen Transformator angeschlossenen PQBus nicht korrekt.
    - \* Fehlerhafte Spannungsangaben führten zu einer falschen Fehlermeldung.
  - Autolayout-Funktion fehlt in der Menubar.
  - Nachdem verschiedene Views entfernt wurden, blieb die Anzeige in der Menubar als „aktiv“ gekennzeichnet.
  - Benennung der verschiedenen Views inkonsistent.
  - Verhindern, dass Transformatoren direkt an RefBus-Komponenten angeschlossen werden können.
- Features
    - Ausgabe von Fehlermeldungen zusätzlich in der Konsole loggen.
    - Fehlermeldung zu den unterschiedlichen Spannungsebenen zweier Elemente bedarf zusätzlicher Hervorhebung der betroffenen Elemente.
    - Anzeige der IDs der Elemente nicht ausschließlich über Tooltips gestalten.
    - Implementierung einer Möglichkeit zur Benennung von Leitungen.
    - Darstellung des PQBus irreführend, da Symbol im Modus-Panel und im ScenarioPanel unterschiedlich sind.
    - Automatisches Docken von Konsumenten/ Produzenten/ Prosumenten an PQBus nachdem diese per Drag& Drop auf einen PQBus gezogen wurden.
    - Implementierung einer Start-Konfiguration für die Simulation, bei der wesentliche Parameter vor dem Start noch einmal überprüft und gegebenenfalls verändert werden können.
    - Zusätzliche Meldungen, wenn bestimmte Einstellungen während der Simulation nicht änderbar sind.
    - Implementierung einer Exportfunktion für berechnete Daten von Netzkomponenten wie Branch, PQBus oder Transformer.
    - Veränderte Darstellung der Wirk-, Blind- und Scheinleistung des Branches als Pfeildiagramm.

- Implementierung eines Kontextmenüs.
- Hinzufügen eines Popups mit der Begründung, warum die Simulation nicht gestartet werden konnte.
- Implementierung, dass an Transformatoren angelegte PQBus-Komponenten direkt die korrekten Werte in Abhängigkeit des Transformatortypen erhalten.

Die Auswertung des SUS-Fragebogen hat einen Wert von 72.25 hervorgebracht, der somit eine gute Gebrauchstauglichkeit der Software belegt. Bekräftigt wird dieses Ergebnis durch das insgesamt sehr positive Feedback der Teilnehmer der Studie, die sich unter anderem über Darstellungsform, Funktionsumfang und Benutzerfreundlichkeit geäußert haben.

Aufgrund weiten Fortschritts des Projekts wurde bei der Bearbeitung der Ergebnisse dieser Studie der Fokus auf das Beheben der Fehler gelegt, damit die Software möglichst fehlerfrei ausgeliefert werden kann. Daher wurde von der Implementierung neuer Features zu diesem Zeitpunkt abgesehen.

## 7.3. Performance

### 7.3.1. Evaluation von ForceAtlas 2

Die Optimierung von bestehenden Szenarien durch ForceAtlas 2 ist ein Prozess, der das System an die Grenzen seiner Rechenleistung bringen kann. Der Abschluss einer Berechnung nimmt eine Zeitspanne in Anspruch, die evaluiert wurde und den Nutzer auf zu erwartende Preis-Leistungs-Verhältnisse aufmerksam macht. Im vorliegenden Testfall soll eine Optimierung nach ForceAtlas 2 in 50 Iterationen durchgeführt werden. Hierfür wird gemessen, wie lange diese Iterationen für Graphen mit steigender Anzahl von Knoten und Kanten zur Ausführung benötigen. Die Größe der Graphen steigt beginnend bei 50 Elementen um jeweils 50 bis 500 zu testenden Elementen an. Ebenfalls wird verglichen, ob die graphische Oberfläche oder der Algorithmus an sich performancekritisch zu betrachten ist. Die Iterationsintervalle werden einmal mit Einblendung der GUI, einmal ohne GUI allein mit der Betrachtung des arbeitenden Algorithmus getestet, um diese Fragestellung zu beantworten. Unter folgenden Konditionen wurden die Tests durchgeführt:

- Gemessen wird die Differenz von dem Endzeitpunkt minus dem Startzeitpunkt.
- Ein gestarter Durchlauf endet nach insgesamt 50 Durchläufen des Algorithmus.

- Der Rahmen beläuft sich auf 0 bis 500 Elemente in der Szenario-Ansicht.
- Testwerte wurden in einem Abstand von 50 Elementen je Messung aufgenommen.
- Es wird mit Anzeige der GUI getestet und ohne diese.
- Wird mit Anzeige der GUI getestet, wird zur Vorbereitung einer Optimierung das Fenster bezogen auf die Größe neu skaliert und angepasst.
- Wird mit Anzeige ohne GUI getestet, wird zur Vorbereitung einer Optimierung das Fenster bezüglich des Szenarios ausser Sichtweite geschoben.

• Daten des Testrechners:

**Rechner** Schenker XMG A522 Advanced Gaming Notebook

**Prozessor** Intel Core i7-3740QM “Ivy-Bridge” Quad-Core Prozessor mit 4 x 2.70GHz, 6MB Cache, Hyper Threading und Turbo Boost bis zu 3.70GHz

**Arbeitsspeicher** 16GB DDR3 mit 1600MHz (2x 8GB verbaut, max. 24GB möglich)

**Grafikkarte** NVIDIA GeForce GTX 660M Grafikkarte mit 2GB dediziertem GDDR5 Speicher und NVIDIA Optimus Technologie

**Betriebssystem** Microsoft Windows 7 Professional (SP1)

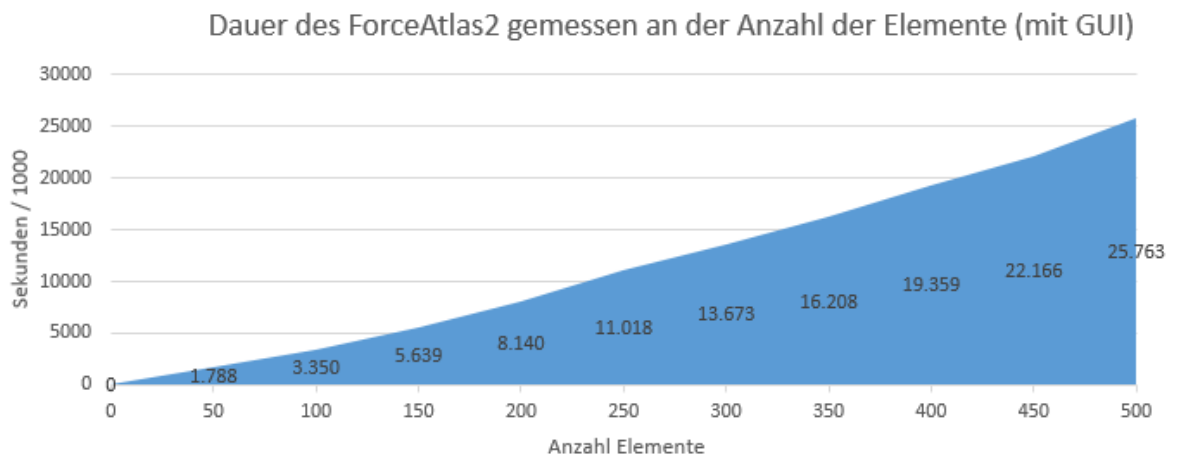


Abbildung 7.2.: Performance der ForceAltas 2-Optimierung mit GUI-Anzeige

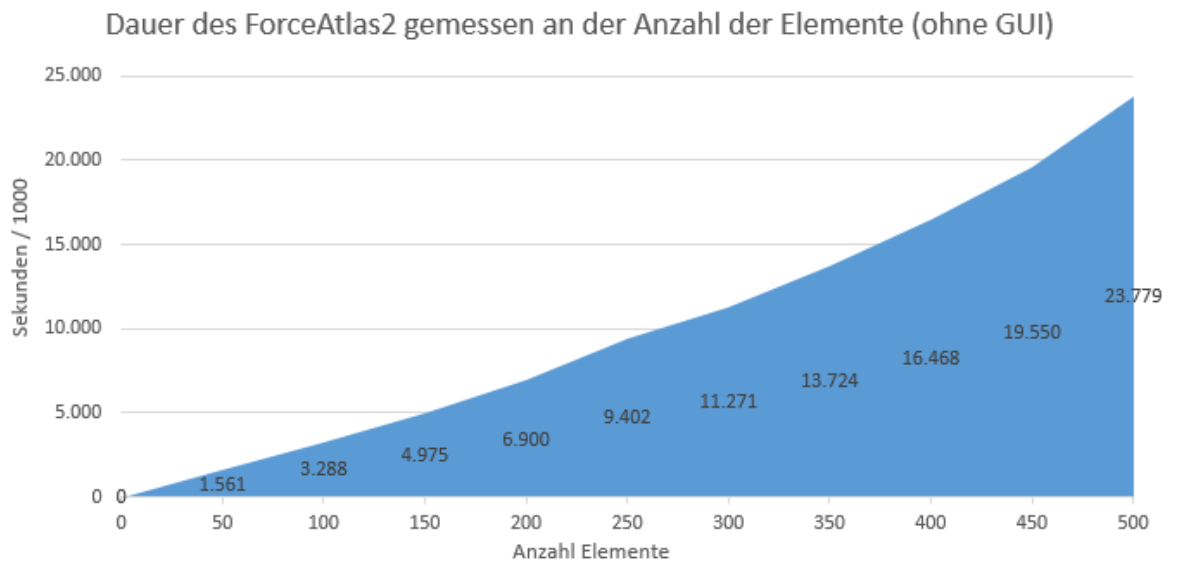


Abbildung 7.3.: Performance der ForceAltas 2-Optimierung ohne GUI-Anzeige

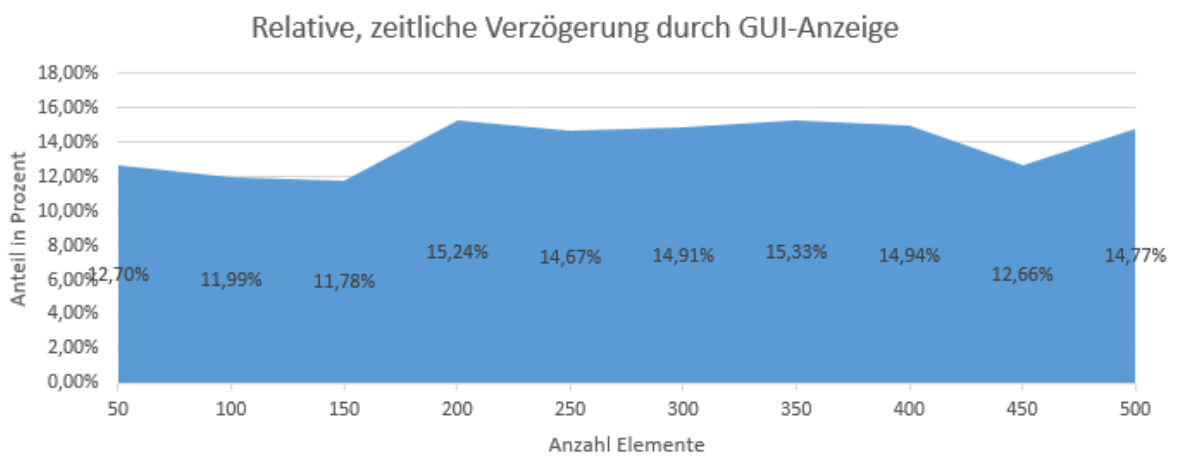


Abbildung 7.4.: Zeitliche Verzögerung von FA2 durch Einblendung der grafischen Oberfläche

Aus den Abbildungen 7.2 und 7.3 wird ersichtlich, dass für dieses System eine leicht exponentielle Steigung vorliegt. Zu beachten ist hierbei, dass bei einem langsameren System die exponentielle Steigung stärker ins Gewicht treten kann. Abbildung 7.4 zeigt, dass die relative Verzögerung von Graphenoptimierungen durch die

Einblendung einer graphischen, angepassten Oberfläche bei cirka zehn bis 15 Prozent einzuordnen ist. Folglich entfällt mit 85 bis 90 Prozent der Auslastung ein Großteil auf den eigentlichen Algorithmus, sodass bezüglich einer geplanten Performanceverbesserung die Verschlinkung der graphischen Oberfläche den kleineren Teil ausmacht. Generell festzuhalten bleibt dennoch, dass aufgrund der exponentiellen Steigung der Zeit, die Software für Graphenoptimierungen kleineren Ausmaßes geeigneter ist.

Mit einem maximalem Fehler von 1.02 Sekunden und einer Standardabweichung von 0.22 Sekunden lässt sich diese Funktion des GUI-Tests ungefähr wie folgt abbilden:

$$y = 4.54e - 5 * x^2 + 0.0308 * x \quad (7.1)$$

Die Funktion ohne GUI-Abbildung entspricht mit maximalem Fehler von und einer Standardabweichung von Sekunden:

$$y = 5.71e - 8 * x^2 + 0.0322 * x \quad (7.2)$$

Angenähert wurde diese Formel durch ein klassisches Brute-Force-Verfahren (Software: Eureka).

### 7.3.2. Evaluation der Element-Selektion

Die Verschiebung mehrerer Elemente in bestehenden Szenarien durch Selektion ist ein Prozess, der das System an die Grenzen seiner Rechenleistung bringen kann. Nachfolgend wird evaluiert, wie sich die Rechenleistung auf die Performance des Systems auswirkt. Die durchgeführte Evaluierung wurde unter folgenden Konditionen durchgeführt:

- Der Rahmen beläuft sich auf 1 bis 140 Elemente in der Szenario-Ansicht. Dabei wurde nach jedem Hinzufügen von Elementen der Force-Atlas angewendet.
- Testwerte wurden in einem Abstand von 20 je Messung aufgenommen.

- Daten des Testrechners:

**Rechner** HP Pavilion dv6 Notebook PC

**Prozessor** Intel Core i5-2410M CPU @ 2.30GHz

**Arbeitsspeicher** 8,00GB

**Grafikkarte** Radeon HD 6490M Grafikkarte

**Betriebssystem** Microsoft Windows 7 Home Premium, 64 Bit-Betriebssystem

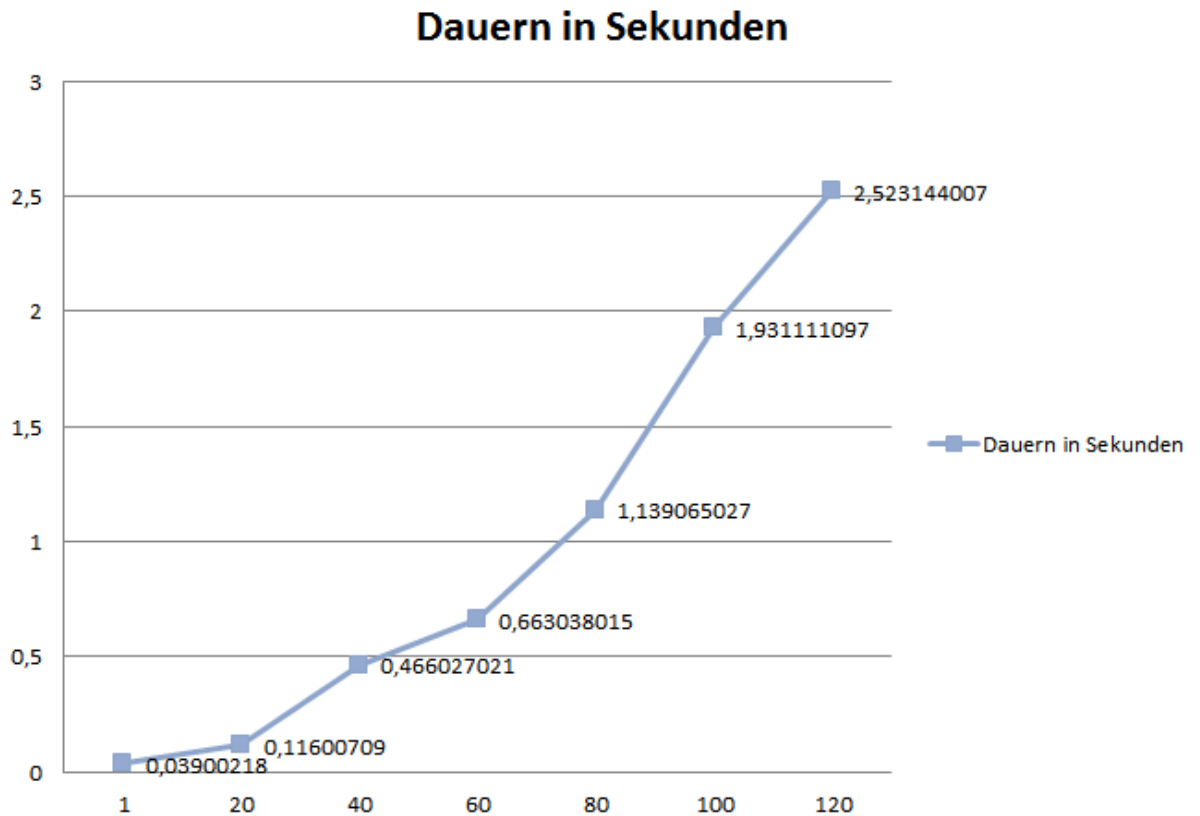


Abbildung 7.5.: Performance der mehrfachen Verschiebung

Aus Abbildung 7.5 wird ersichtlich, dass für dieses System eine relativ lineare Steigung vorliegt. Bei der Evaluierung der Verschiebung mit 140 Elementen erfolgte ein Systemabsturz.



## 8. Projektabschluss

Im folgenden Kapitel wird ein Gesamtfazit zum Projekt gezogen und die Projektdokumentation mit einem detaillierten Ausblick zu Verbesserungsmöglichkeiten sowie Möglichkeiten zur Weiterentwicklung abgeschlossen.

### 8.1. Fazit

Das Ziel dieser Arbeit war die Entwicklung einer Software zur Visualisierung und Simulation von Smart Grids in einem 12 Mitglieder umfassenden Team. Das folgende Fazit bezieht sich auf die Punkte Teamarbeit und Projektverlauf während der Projektphase.

Die Teamarbeit hat rückblickend auf das Projekt sehr gut funktioniert. Für die reibungslose Zusammenarbeit war zweifellos die ständige Kommunikation während des gesamten Projektverlaufs ausschlaggebend, da dadurch sofort wichtige Details, Ideen oder Probleme besprochen und gemeinsam gelöst werden konnten.

Außerdem waren die wöchentlichen Treffen (Weekly Scrum) für die Entstehung einer Gruppendynamik sehr förderlich, da diese ebenfalls eine verstärkte Kommunikation ermöglicht haben. Bei diesen Treffen wurden unter anderem der aktuelle Stand besprochen sowie weitere Aufgaben verteilt. Außerdem haben die Mitglieder sich gegenseitig über ihr Wochenpensum und die jeweiligen Projektzwischenstände informiert. Die wöchentlichen Treffen trugen zur Transparenz bei und sorgten dafür, dass alle gleichermaßen am Projekterfolg beteiligt waren. Durch die selbstständige Mitarbeit und die hohe Motivation des Teams, einem sehr guten Konfliktmanagement sowie einer exzellenten projektinternen Planung und Organisation, konnten die vorgenommenen Ziele ohne Verzögerung erreicht werden.

Für den Projektverlauf war das ausführliche Product-Backlog ein zentrales Element. In diesem Dokument wurden alle Anforderungen detailliert aufgegliedert, sodass eine realistische Einschätzung des Arbeitsaufwandes ermöglicht wurde. Anhand dessen konnte ein Zeitplan erstellt werden, der dafür sorgte, dass vorgenommene Ziele ohne Verzug erreicht wurden. Dennoch gab es während des Projektverlaufs immer wieder Hürden und Hindernisse, die den Projektfortschritt hätten gefährden können. Durch eine schnelle und effektive Reaktion auf diese, konnte die Fertigstellung des Produkts zum genannten Endtermin trotzdem gewährleistet werden. Dazu ge-



hörten beispielsweise neue Anforderungen der Betreuer oder sich aus den Usability-Studien ergebende zusätzliche Anforderungen. Darüber hinaus wurden ein umfangreiches Refactoring sowie Performanceoptimierungen und ein verspäteter Projektstart aufgrund der Seminararbeiten problemlos bewältigt.

Des Weiteren war die Rollenverteilung der Mitglieder für die gut funktionierende Teamarbeit von hoher Bedeutung. So gab es Product-Owner, Scrum-Master oder Dokumenten-Manager und somit für jedes wichtige Themengebiet zwei *Spezialisten*, die sich mit dem Thema auseinandergesetzt haben. Abschließend lässt sich sagen, dass das gesamte Team durch dieses umfangreiche Projekt viele positive Erfahrungen sammeln konnte, sei es in der Planung von größeren Projekten, in der Zusammenarbeit mit anderen Teammitgliedern oder auch die Erweiterung der individuellen Programmierfähigkeiten.

## 8.2. Ausblick

Im Rahmen unserer Projektgruppe haben wir uns ebenfalls Gedanken über verschiedene Möglichkeiten zur Weiterentwicklung der Simulations- und Visualisierungssoftware Maverig gemacht. Aufgrund der hohen Anzahl an Anforderungen und der zeitlichen Begrenzung konnten nicht alle Aspekte umgesetzt werden. Daher bieten wir hiermit einen Ausblick für andere Projektgruppen, Masterarbeiten oder ähnliches.

Die Ideen zur Weiterentwicklung stammen zum einen aus unseren wöchentlichen Treffen, bei denen wir in enger Zusammenarbeit mit unseren Betreuern Erweiterungsmöglichkeiten diskutiert und dokumentiert haben und zum anderen aus dem Feedback der Produktzwischenpräsentation sowie aus den beiden bereits durchgeführten Usability-Studien.

Im Folgenden werden exemplarisch ausgewählte Erweiterungsmöglichkeiten dargestellt. So ist beispielsweise ein regelbasiertes, mehrfaches Hinzufügen und Verbinden von Komponenten vorstellbar. Dies würde das Anlegen von umfangreichen Szenarien erleichtern. Mit dieser Funktion wäre es mithilfe eines Einstellungsfensters möglich, dem Szenario beliebig viele Komponenten mit gleichen Eigenschaften hinzuzufügen. Im Abschnitt 8.2.1 finden Sie ein ausführliches Konzept für die Realisierung. Weiterhin wäre es denkbar Maverig als Converter zu verwenden, sodass sich unter anderem auch andere Formate, außer PyPower, laden und zu einem bestehenden Szenario hinzufügen lassen.

Eine weitere Ausbaumöglichkeit wäre eine Export-Funktion für die Daten von einem Branch, Node oder Transformer, damit diese für weitere Analysezwecke genutzt werden könnten. Ebenfalls könnten die bereits ausgegebenen Fehlermeldungen deutlicher hervorgehoben werden, sodass der Benutzer den Fehler noch einfacher und

schneller lokalisieren und beheben kann. Im Abschnitt 8.2.2 wird ein Konzept für die Integration von Kontrollmechanismen vorgestellt. Weitere Erweiterungsmöglichkeiten können der Liste im Abschnitt 8.2.3 entnommen werden.

### 8.2.1. Konzept für mehrfaches Erstellen und Verbinden von Elementen

#### Mehrfaches Hinzufügen von Elementen über einen Wizard

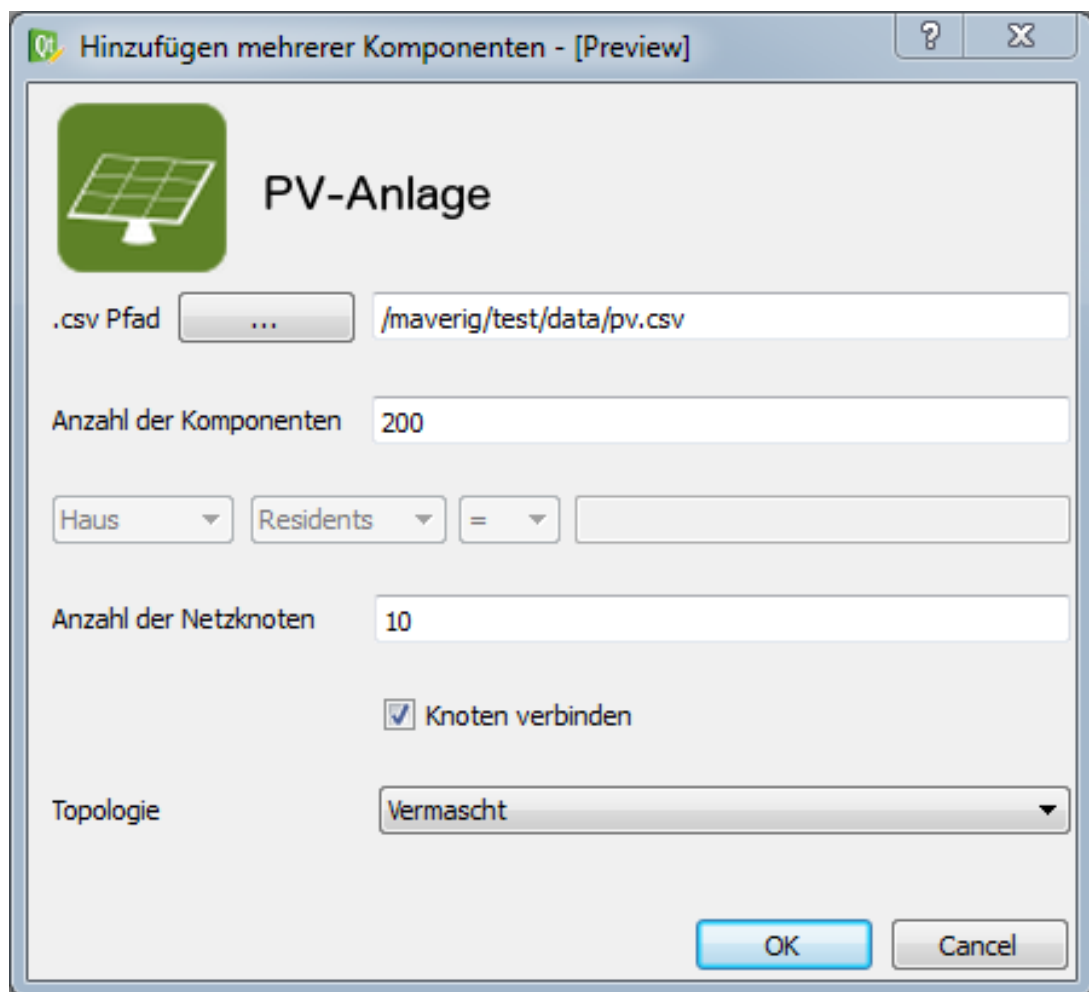


Abbildung 8.1.: Mockup: Hinzufügen mehrerer Elemente

Das Erstellen mehrerer Elemente könnte darüber realisiert werden, dass in den entsprechenden Kategorien ein zusätzlicher Button eingefügt wird, der einen Dialog

öffnet. Dieser Dialog enthält Abfragen darüber, wie viele Elemente der ausgewählten Komponente hinzugefügt werden sollen sowie die Anzahl der zu erzeugenden Knoten. Die Anzahl der Knoten ist notwendig, falls zuvor kein Knoten im Szenario ausgewählt wurde. Wenn kein Knoten im Szenario ausgewählt bzw. markiert wurde, können die Komponenten gleichmäßig auf die Anzahl der neu erstellten Knoten aufgeteilt werden. Alternativ würden alle erzeugten Elemente an dem selektierten Knoten erstellt werden. Sollten mehrere Knoten selektiert worden sein, wird die angegebene Anzahl der Komponenten auf diese verteilt. Zusätzlich kann neuen Knoten bei der Erzeugung eine auswählbare Topologie zugeordnet werden. Dabei werden die Knoten gemäß der ausgewählten Topologie miteinander verknüpft und die Komponenten wiederum gleichmäßig aufgeteilt. Dieser Dialog kann alternativ einen Filter enthalten, in dem über die Angabe von Bedingungen eine entsprechende Aktion ausgeführt werden kann.

*Beispiel: Wenn: Haus | Mit: Mindestens 3 Haushalten | Dann: Füge PV-Anlage hinzu.*

#### **Mehrfaches Erstellen von Elementen über Szenarioimport**

Eine weitere Möglichkeit viele Komponenten hinzuzufügen besteht darin Szenarien importieren zu können. Bislang können lediglich Szenarien geladen werden, wobei dabei das aktuelle Szenario überschrieben wird. Beim Importieren werden ein oder mehrere bereits gespeicherte Szenarien in das aktuelle Szenario geladen.

#### **Mehrfaches Erstellen von Elementen durch CSV**

Als dritte Alternative bietet sich ein zusätzlicher Button an, hinter dem sich ein frei definierbarer oder fest vorgegebener Satz an Elementen verbirgt. Unter Angabe des Pfads der .csv-Datei können die dort enthaltenen Daten aufsummiert und simuliert werden.

### **8.2.2. Konzept Heat-Werte der einzelnen Elemente**

Die Heat-Werte einer jeden Komponente haben Einfluss auf die jeweilige Darstellung während der Simulation, bspw. ab welchem Wert die Komponente sich in einem kritischen Zustand befindet (z.B. Überlast).

Aktuell erfolgt in *maverig.models.model* eine konkrete Berechnung abhängig von der Komponente und ihrer Attribute. Diesem errechneten Heat-Wert wird anschließend in *maverig.presenter.group\_presenter* ein entsprechender Farbwert zugeordnet, welcher zur Visualisierung im Szenario genutzt wird (bspw. >80% gelb, >100% rot). Für jede Komponente wird eine andere Berechnung des Heat-Wertes vorgenommen, als Beispiel wird hier die Interpretation der Spannungsabweichung aufgezeigt:

Bei der Errechnung des Heat-Wertes für die Spannungsabweichung betrachten wir während der Simulation den Anfangs- und Endpunkt einer Leitung(Branch). Mosaik stellt uns die Nennspannung sowie die aktuelle anliegende Spannung als Parameter dieser Endpunkte bereit. Auf Basis dieser Werte können wir die Spannungsabweichung in Prozent ermitteln(Heat-Wert). Für eine Zuordnung der Farbe für diesen Wert wird die Europeanorm EN50160 (*Merkmale der Spannung in öffentlichen Elektrizitätsversorgungsnetzen*) zugrunde gelegt[EN11]. In Deutschland wird diese Europeanorm in der **DIN EN 501060** abgebildet. Sie erlaubt eine maximale Spannungsdifferenz von +/- 10% der Nennspannung in Elektrizitätsversorgungsnetzen, sodass wir auf dieser Basis die betroffene Komponente einfärben. Konkret bedeutet dies, dass eine Abweichung von +/- 10% während der Simulation rot eingefärbt wird.

Wie man an diesem Beispiel erkennt, erfolgt z.Z. eine einzelne Betrachtung der vorhandenen Komponenten, manuell im Model und anschließend im Grouppresenter. Beim Hinzufügen von neuen Komponenten über den Komponentenassistenten führt dies zu Problemen da hier das Model manuell um die neue Komponente erweitert werden müsste.

In Zukunft wäre es möglich, eine Auswahl der Berechnungs- bzw. Interpretationsarten bereits im Komponentenassistenten abzufragen und anschließend automatisch generieren zu lassen. Auch eine Vorgabe von Berechnungsvarianten für verschiedene Gruppen ist denkbar. In der aktuellen Version von Maverig ist für die Komponente Haushalt bereits das definieren einer 100% Grenze über das Eigenschaftenpanel möglich (Maximale Leistung). In Zukunft könnte dies für weitere Komponenten adaptiert werden.

### 8.2.3. Konzept für Integration von Kontrollmechanismen

Komponenten für Kontrollmechanismen und Datensammler<sup>1</sup> können beliebig viele eingehende Verbindungen aufweisen und werden aufgrund der damit verbundenen Visualisierungskomplexität derzeit noch nicht in Maverig unterstützt. Im Folgenden werden bereits bestehende Funktionen und Ansätze zur Erweiterung von Maverig aufgezeigt, um vielfach eingehende Verbindungen vollständig zu unterstützen. Diese sind unter anderem notwendig bei Kontrollstrategien.

Komponenten- und Simulatorbeschreibungen von Kontrollmechanismen zur Einbindung in Maverig können bereits entsprechend dem Mosaik-Kontrollmechanismus-Beispiel<sup>2</sup> wie folgt erstellt werden:

<sup>1</sup>Datensammler mosaik-hdf5: <https://bitbucket.org/mosaik/mosaik-hdf5>

<sup>2</sup>Mosaik Tutorial zu Kontrollmechanismen: <http://mosaik.readthedocs.org/en/latest/>

```

1 {
2   "creation_time": "2015-02-28T21:26:34",
3
4   "sim_model": "ExampleCtrl.Agent",
5
6   "type": ["Component", "ExampleCtrl", "Controller", "Agent"],
7   "category": "Agents",
8   "tooltip": "Agent",
9
10  "icon": "agent.svg",
11  "drawing_mode": "icon",
12
13  "docking_ports": {
14    "0": {
15      "in" : ["Component"]
16    }
17  },
18
19  "on_sim_init": null,
20  "on_set_param": null,
21
22  "published_params": [],
23  "published_attrs": [],
24
25  "params": {},
26
27  "async_requests": True,
28
29  "attrs": {
30    "val_in": {
31      "in": ["val"]
32    }
33  }
34 }

```

**Listing 8.1:** Komponentenbeschreibung für einen Agenten

```

1 {
2   "name": "ExampleCtrl",
3   "starter": "python",
4   "address": "controller:Controller",
5   "params": {},
6   "on_sim_init_parents": null
7 }

```

**Listing 8.2:** Simulatorbeschreibung für einen Kontrollmechanismus

An der Anschlussstelle "0" können gemäß Listing 8.1 beliebige Komponenten eingehen, da jede Komponente den Typ "Component" enthält. Außerdem werden eingehende Verbindungen gemäß der Komponentenbeschreibung mit dem Parameter `async_requests=True` belegt. Auf diese Weise kann der Kontrollmechanismus im Simulationsschritt eingehende Elemente über asynchrone Methodenaufrufe steuern, während er über das Attribut "val\_in" die aktuellen Werte des Attributs "val" der

eingehenden Elemente empfängt.

```

1 def connect_houses_to_agent(model, agent_elem_id):
2     for elem_id in model.elements:
3         if 'House' in model.get_component(elem_id)['type']:
4             model.dock([elem_id, '0'], [agent_elem_id, '0'])

```

**Listing 8.3:** Häuser mit Agent verbinden

Das mehrfache Verbinden von Anschlussstellen ist auf Modellebene über die Funktion `model.dock(from_ep, to_ep)` kein Problem. So könnten beispielsweise wie in Listing 8.3 alle Häuser an einen Agenten andockt werden. Bezüglich der automatischen Docking-Validierung erlauben Häuser in ihrer Komponentenbeschreibung `maverig.data.components.CSV.House.json` ausgehende Verbindungen nur an der Anschlussstelle "1". Ausgehende Verbindungen zu Agenten können durch folgende Änderung erlaubt werden:

```
"docking_ports": {"0": {"out" : ["Agent"]}, "1": ...}.
```

In der View sind Dockings derzeit jedoch so umgesetzt, dass miteinander verbundene Elemente immer gleichzeitig verschoben werden. In diesem Fall sollen jedoch alle verbundenen Elemente unabhängig voneinander verschoben werden können. Dieses Problem könnte durch spezielle Verbindungslinien mit weiteren Anschlussstellen oder eine Sonderbehandlung bei den Dockings in `maverig.presenter.group_presenter.abstractGroupPresenter.py` umgesetzt werden.

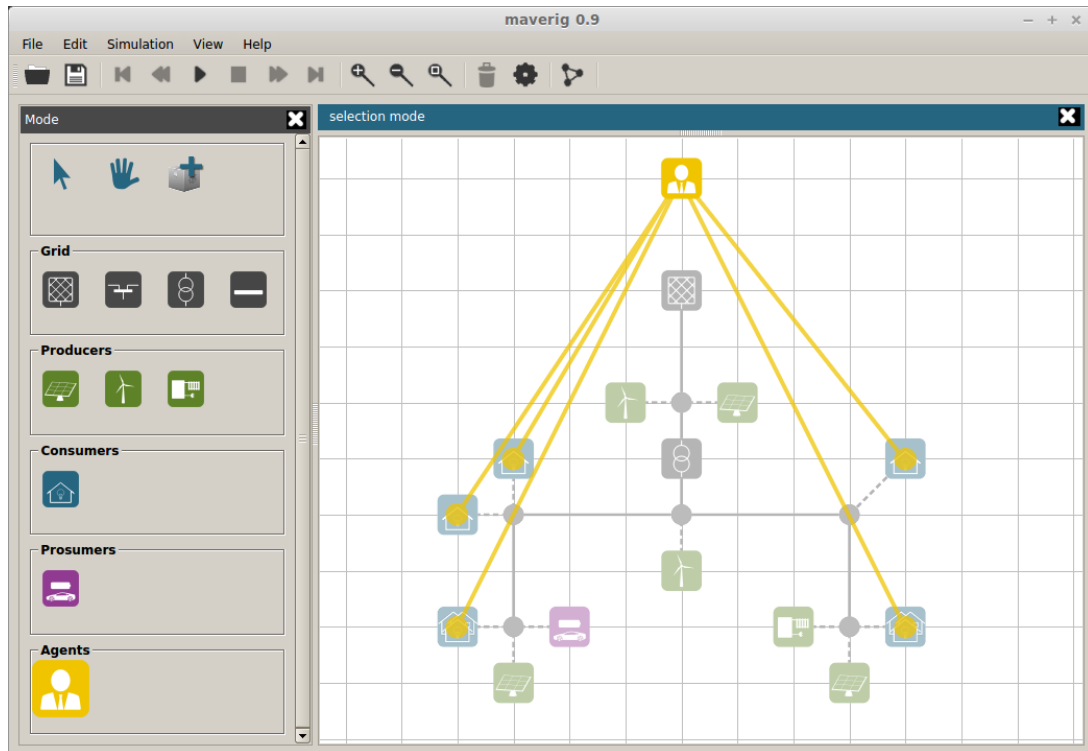


Abbildung 8.2.: Mockup: Visualisierung von Agentenverbindungen

In Abbildung 8.2 befindet sich ein Vorschlag zur Visualisierung eingehender Agentenverbindungen, um eine Unterscheidung zu den gestrichelten ausgehenden Verbindungslinien der einzelnen Icons zu verdeutlichen. Außerdem könnten die neuen Verbindungslinien nur bei einer Auswahl des Agenten sichtbar sein, um die Übersichtlichkeit der Kompositionsansicht weiterhin zu gewährleisten. Die Verbindungslinien könnten beispielsweise durch einen „Verbindungslinien-Zeichnen-Modus“ oder einen „Wizard für mehrfaches Verbinden“ über das Kontextmenü eines Agenten eingefügt werden.

#### 8.2.4. Weiterentwicklungsmöglichkeiten für Maverig

1. Realisierung einer 3D-Visualisierung.
2. Automatische Visualisierung von PyPower.json.
3. Optimierung des Auto-Layout im Bezug auf Transformator und Grid.
4. Abfangen unzulässiger Intervallstufen bei der Zeiteinstellung.
5. Deutliche Hervorhebung von Fehlerkomponenten.





6. Umspannung von nur einer Ebene durch Transformatoren.
7. Realisierung von Fehlermeldungen mit visuellem Feedback.
8. Darstellung von Knoten(PQBus) als Icon sowie Anpassung der Umsetzung im Szenario.
9. Direktes Andocken von Elementen bei Drag&Drop.
10. Anzeige eines Konfigurationsfensters mit Einstellungen vor dem Start des Simulators.
11. Feedback während der Simulation, bei nicht änderbaren Einstellungen.
12. Darstellung der Schein- und Wirkungsleistung von Leitungen(Branch) als Pfeildiagramm.
13. Popup bei nicht möglichem Simulationsstart.
14. Automatische Validierung von Knoten, die an einem Transformator mit falscher Spannung angeschlossen sind.

## Literaturverzeichnis

- [AK+12] APPELRATH, H.-J., KAGERMANN, H. ET AL.: *Future Energy Grid, Migrationspfade ins Internet der Energie*, aca- tech – Deutsche Akademie der Technikwissenschaften, 2012, Seite 99, Abbildung 11: Systemmodell des „European Electricity Grid Initiative and Implementation plan“ (angelehnt an Darstellung der AG 2 des IT-Gipfels)
- [Baro4] SKIZZE DER FUNKTIONSWEISE EINES WÄRMEKRAFTWERKES, 2004, [http://commons.wikimedia.org/wiki/File:W%C3%A4rmekraftwerk\\_schematisch.png](http://commons.wikimedia.org/wiki/File:W%C3%A4rmekraftwerk_schematisch.png), Stand 10.03.2015
- [BDI12] DIE BUNDESBEAUFTRAGTE FÜR DEN DATENSCHUTZ UND DIE INFORMATIONSFREIHEIT: *Orientierungshilfe zum datenschutzgerechten Smart Metering*, 2012, [http://www.bfdi.bund.de/SharedDocs/Publikationen/Entschliessungssammlung/DSBundLaender/Entschliessung\\_SmartMeter.pdf;jsessionid=80DE20098DCE928DCEB0E1977D131FA0.1\\_cid344?\\_\\_blob=publicationFile](http://www.bfdi.bund.de/SharedDocs/Publikationen/Entschliessungssammlung/DSBundLaender/Entschliessung_SmartMeter.pdf;jsessionid=80DE20098DCE928DCEB0E1977D131FA0.1_cid344?__blob=publicationFile), Stand 11.05.2014
- [BEW13] BUNDESVERBAND DER ENERGIE- UND WASSERWIRTSCHAFT E.V.: *Erneuerbare Energien und das EEG*, 2013, [https://www.bdew.de/internet.nsf/id/17DF3FA36BF264EBC1257B0A003EE8B8/\\$file/Foliensatz\\_Energie-Info-EE-und-das-EEG2013\\_31.01.2013.pdf](https://www.bdew.de/internet.nsf/id/17DF3FA36BF264EBC1257B0A003EE8B8/$file/Foliensatz_Energie-Info-EE-und-das-EEG2013_31.01.2013.pdf), Stand 11.05.2014
- [BNA11] BUNDESNETZAGENTUR: „Smart Grid“ und „Smart Market“, *Eckpunktpaier der Bundesnetzagentur zu den Aspekten des sich verändernden Energieversorgungssystems*, Bonn, 2011, [http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Energie/Unternehmen\\_Institutionen/NetzzugangUndMesswesen/SmartGridEckpunktepapier/SmartGridPapierpdf.pdf?\\_\\_blob=publicationFile&v=2](http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Energie/Unternehmen_Institutionen/NetzzugangUndMesswesen/SmartGridEckpunktepapier/SmartGridPapierpdf.pdf?__blob=publicationFile&v=2), Stand 09.05.2015
- [Cle13] CLEMENS, J.: *Energieverbrauch sinkt, Stromverbrauch steigt*, Axel Springer SE, 2013, <http://www.welt.de/dieweltbewegen/sonderveroeffentlichungen/article120919628/Energieverbrauch-sinkt-Stromverbrauch-steigt.html> Stand 11.05.2014

- [DBU08] DEUTSCHER BUNDESTAG: *Gesetz zur Öffnung des Messwesens bei Strom und Gas für Wettbewerb*, Bonn, 2008, BgBl. I 2008, 1790 <http://www.bmwi.de/Dateien/Energieportal/PDF/gesetz-oeffnung-messwesen,property=pdf,bereich=bmwi2012,sprache=de,rwb=true.pdf> Stand 11.05.2014
- [FEK14] FRAUNHOFER-INSTITUT FÜR EINGEBETTETE SYSTEME UND KOMMUNIKATIONSTECHNIK ESK: *Smart Grid Communications 2020*, [http://www.esk.fraunhofer.de/content/dam/esk/de/documents/SmartGrid\\_Studie\\_final-web.pdf](http://www.esk.fraunhofer.de/content/dam/esk/de/documents/SmartGrid_Studie_final-web.pdf) Stand 11.05.2014
- [ELK15a] ELEKTRONIK KOMPENDIUM: *Elektrischer Strom*, <http://www.elektronik-kompodium.de/sites/grd/0110203.htm>, Stand 16.2.2015
- [ELK15b] ELEKTRONIK KOMPENDIUM: *Elektrische Spannung*, <http://www.elektronik-kompodium.de/sites/grd/0201101.htm>, Stand 16.2.2015
- [ELK15c] ELEKTRONIK KOMPENDIUM: *Elektrische Leistung*, <http://www.elektronik-kompodium.de/sites/grd/0201114.htm>, Stand 16.2.2015
- [ELK15d] ELEKTRONIK KOMPENDIUM: *Elektrische Arbeit*, <http://www.elektronik-kompodium.de/sites/grd/0306111.htm>, Stand 16.2.2015
- [ELK15e] ELEKTRONIK KOMPENDIUM: *Drehstrom - Dreiphasenwechselstrom*, <http://www.elektronik-kompodium.de/sites/grd/1006061.htm>, Stand 16.2.2015
- [EN11] EN 50160: *Merkmale der Spannung in öffentlichen Elektrizitätsversorgungsnetzen*, <http://www.b-w.at/download/category/29-b-w-wissensdatenbank.html?download=212:en50160>, Stand 04.03.2015
- [Fob07a] FONKENG, BIRKA: *Layout- und Filterverfahren zur Graphdarstellung in GroIMP*, Brandenburgische Technische Universität Cottbus, Diplomarbeit - 2007, S.3.
- [Fob07b] Wiederholtes Zitat: FONKENG, BIRKA: a.a.O., S.3f
- [GEP11] GEPHI.ORG: *ForceAtlas2, the new version of our home-brew Layout*, <https://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout/>, Stand 06.05.2014
- [GNU14a] GNU OPERATING SYSTEM: *GNU GENERAL PUBLIC LICENSE*, <http://www.gnu.org/licenses/gpl-3.0.en.html>, Stand 26.01.2015

- [GNU14b] GNU OPERATING SYSTEM: *GNU LESSER GENERAL PUBLIC LICENSE*, <http://www.gnu.org/licenses/lgpl-3.0>, Stand 26.01.2015
- [KK93] KOPFMÜLLER, K., KOHN, G.: *Theoretische Elektrotechnik und Elektronik*, 14. Auflage, Springer 1993
- [Kla14a] KLASSEN, GERRIT: *Motivationen und Algorithmen zur Erstellung von Grafklayouts* Carl von Ossietzky Universität Oldenburg, Seminararbeit - 2014, S.11f.
- [Kla14b] Wiederholtes Zitat: KLASSEN, GERRIT: a.a.O., S.7f
- [KS+09] KNAB, S., STRUNZ, K., LEHMANN, H.: *Smart Grid: The Central Nervous System for Power Supply - New Paradigms, New Challenges, New Services*, 2009, Scientific Series of the Innovation Centre Energy at the Technische Universität Berlin, Vol. 2, University Press, Berlin, Germany. Available at SSRN: <http://ssrn.com/abstract=1531655>
- [Leh13a] LEHNHOF, SEBASTIAN: *Aufbau und Betrieb elektrischer Energieversorgungsnetze*, Carl von Ossietzky Universität Oldenburg, Präsentation - 2013, S.20
- [Leh13b] Wiederholtes Zitat: LEHNHOF, SEBASTIAN: a.a.O., S.19.
- [Lip14a] LIPINSKI, K.: *Smart Home*, DATACOM Buchverlag GmbH, 2014, <http://www.itwissen.info/definition/lexikon/E-Home-eHome-electronic-home.html> Stand 10.05.2014
- [Lip14b] LIPINSKI, K.: *Smart Meter*, DATACOM Buchverlag GmbH, 2014, <http://www.itwissen.info/definition/lexikon/smart-meter-Intelligenter-Zaehler.html> Stand 10.05.2014
- [Pas14] PASCHOTTA, D.R.: *Energie Lexikon*, <http://www.energie/lexikon.info>, Stand 12.5.2014
- [Pas14b] PASCHOTTA, R.: *Hochspannungsleitung*, <http://www.energie-lexikon.info/hochspannungsleitung.html> Stand 11.05.2014
- [RR+13] RICHTER, K., SCHARF, D. ET. AL. *IT-Handbuch*, 8. Auflage, Westermann 2013
- [Sch12a] SCHWAB, A.J.: *Elektroenergiesysteme*, Heidelberg, neu bearbeitete und erweiterte Auflage 2012, S.79.
- [Sch12b] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.233-236.
- [Sch12c] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.436-439.

- [Sch12d] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.439-440
- [Sch12e] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.519-531.
- [Sch12f] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.22-23.
- [Sch12g] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.531-539.
- [Sch12h] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.512-513.
- [Sch12i] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.514-516.
- [Sch12j] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.193.
- [Sch12k] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.513.
- [Sch12l] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.514.
- [Sch12m] Wiederholtes Zitat: SCHWAB, A.J.: a.a.O., S.515.
- [SWI10] SWISSGRID *Transmission Code 2010*, [http://www.swissgrid.ch/dam/swissgrid/experts/transmission\\_code/TC\\_2010\\_de.pdf](http://www.swissgrid.ch/dam/swissgrid/experts/transmission_code/TC_2010_de.pdf), Stand 12.5.2014
- [QT12] QT-PROJECT.ORG: *About PySide*, <http://qt-project.org/wiki/About-PySide>, Stand 07.02.2015
- [Wago3] WAGNER, H.-F.: *Struktur des deutschen Stromnetzes*, 2012, <http://www.weltderphysik.de/gebiete/technik/energie/speichern-und-transportieren/strom/netzstruktur/> Stand 09.05.2014

## A. Anhang

### A.1. Benutzerhandbuch

#### a. Einführung

Maverig ist eine grafische Oberfläche zur Erstellung und Visualisierung einer Smart-Grid Simulation. Maverig teilt sich bei der Bedienung in den Kompositions-Modus und den Simulations-Modus auf. Im Kompositions-Modus wird im ersten Arbeitsschritt ein Smart-Grid-Szenario erstellt, welches anschließend im Simulations-Modus ausgeführt werden kann, inklusive Überwachung wichtiger Parameter. Maverig nutzt hierzu die Simulatoren von Mosaik.

#### b. Aufbau des Benutzerhandbuches

Dieses Handbuch soll einen grundlegenden Überblick über die Funktions- und Bedienungsweise von Maverig geben. Hierfür ist das Handbuch in 2 Teile unterteilt. Im ersten Teil des Handbuches wird die Bedienung des Kompositions-Modus zur Erstellung eines Szenarios erläutert. Der Zweite Teil des Benutzerhandbuches beschreibt den Umgang mit dem Simulations-Modus.

#### c. Installation

Maverig unterstützt die Plattformen Linux, OSX und Windows. Zusätzlich wird mindestens Python 3.4 oder neuere Version zur Ausführung benötigt. Um Maverig zu installieren wird der Paketmanager *pip* benötigt, welcher in der Python 3.4 Installation enthalten ist. Die Installation erfolgt über: *pip install maverig*

#### d. Aufbau GUI

In diesem Kapitel wird auf den Aufbau der grafischen Oberfläche von Maverig eingegangen. Hierzu werden die einzelnen Elemente und ihre Funktionsweise detailliert erläutert. Grundsätzlich ist der Aufbau im Kompositions- und Simulationsaufbau sehr ähnlich. Im Kompositions-Modus unterteilt sich die GUI in die Bereiche Menüleiste (1), Toolbar (2), Moduspanel (3), Eigenschaftenpanel (4), Statusbar (5), Szenariopanel (6) und Konsole (7).

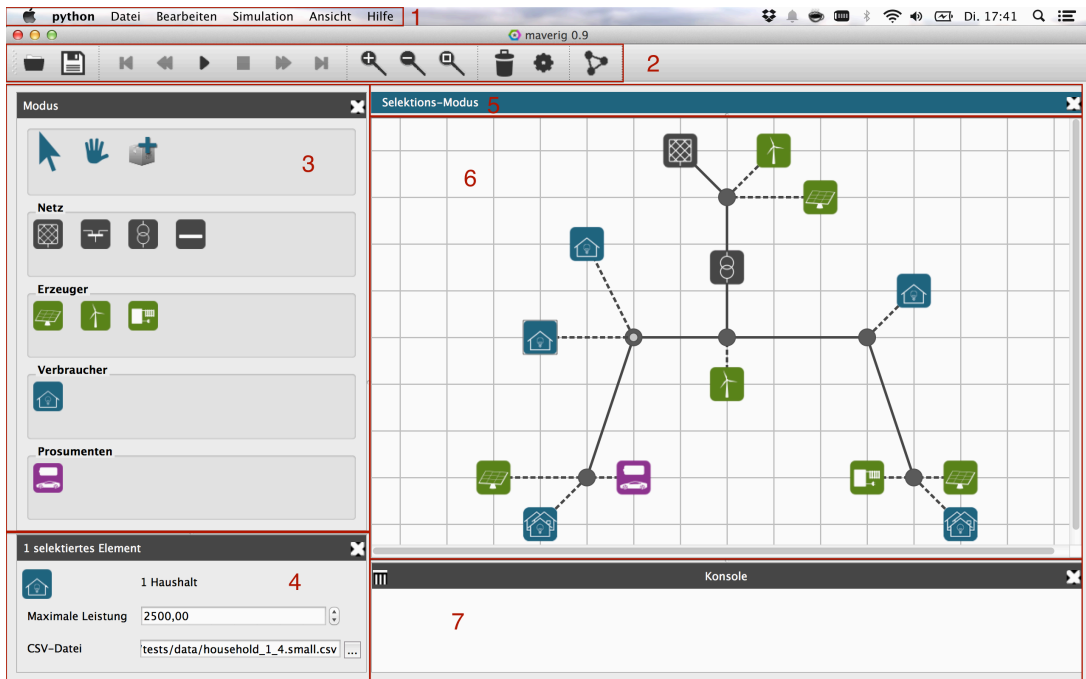


Abbildung A.1.: Übersicht GUI Maverig Komposition

1. *Menüleiste*

Über die Menüleiste können Sie auf verschiedene Funktionen sowie Einstellungsmöglichkeiten von Maverig zugreifen. Viele der Funktionen aus der Menüleiste sind zusätzlich in der Toolbar hinterlegt.

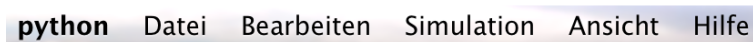


Abbildung A.2.: Maverig Menüleiste

Datei	Neu	Stellt ein neues, leeres Szenario bereit
	Öffnen	Öffnet ein bereits bestehendes Maverig Szenario
	Speichern	Speichert das Szenario
	Speichern unter	Speichert das Szenario unter einem angegebenen Pfad
	Einstellungen	Allgemeine Einstellungen sowie spezifische Simulationseinstellungen von Maverig
	Schließen	Beendet Maverig
Bearbeiten	Rückgängig	Macht den zuletzt ausgeführten Arbeitsschritt im Szenario rückgängig
	Wiederherstellen	Wiederholt den zuletzt ausgeführten Arbeitsschritt im Szenario
	Automatisches Ausrichten	Ordnet die platzierten Komponenten des Szenarios übersichtlich an
	Ausschneiden	Schneidet ausgewählte Elemente aus und fügt sie in die Zwischenablage ein
	Kopieren	Kopiert ausgewählte Elemente und fügt sie in die Zwischenablage ein
	Einfügen	Fügt gespeicherte Elemente aus der Zwischenablage in das Szenario ein
	Löschen	Löscht ausgewählte Elemente aus dem Szenario
	Alle Markieren	Wählt alle Elemente des Szenarios aus
Simulation	Ausführen	Startet die Simulation der Komposition
	Stoppen	Stoppt die Simulation der Komposition
	Pause	Pausiert die Simulation der Komposition
	Zum Start springen	Springt zum Startzeitpunkt der Simulation
	Geschwindigkeit reduzieren	Reduziert die Wiedergabegeschwindigkeit der Simulation
	Geschwindigkeit erhöhen	Erhöht die Wiedergabegeschwindigkeit der Simulation
	Zeit einstellen	Anpassung der Simulationsstart- und Endzeit
	Simulationszeit ändern	Anpassung der Simulationsstart- und Endzeit
	Gehe zu	Springt zu angegebenem Zeitpunkt in der Simulation



Ansicht	Shift-Modus	Aktiviert den Shift-Modus
	Selektions-Modus	Aktiviert den Selektions-Modus
	Raster	Aktiviert oder Deaktiviert das automatische Andocken der Elemente am Raster
	Vergrößern	Vergrößert die Ansicht des Szenarios
	Verkleinern	Verkleinert die Ansicht des Szenarios
	Szenario anpassen	Passt die Ansicht des Szenarios automatisch an
	Ausblenden von Komponenten	Ein- oder Ausblenden von Darstellungskomponenten.
Hilfe	Maverig Hilfe	Öffnet die Maverig Hilfe
	Über Maverig	Stellt Informationen über Maverig, wie z.B. beteiligte Entwickler, bereit.

2. *Toolbar*

Über die Buttons der Toolbar kann auf verschiedene Funktionen von Maverig zugegriffen werden. Zusätzlich sind viele Funktionen aus der Toolbar über die Menüleiste von Maverig zu erreichen.






Abbildung A.3.: Maverig Toolbar

	Öffnen	Öffnet ein bereits bestehendes Maverig Szenario
	Speichern	Speichert das Szenario
	zum Start springen	Springt zum Start des Szenarios
	Geschwindigkeit verringern	Verringert die Wiedergabegeschwindigkeit des Szenarios
	Start	Startet die Simulation der Komposition
	Stop	Stoppt die Simulation der Komposition
	Geschwindigkeit erhöhen	Erhöht die Wiedergabegeschwindigkeit des Szenarios
	zum Ende springen	Springt zum Start des Szenarios
	Hereinzoomen	Vergrößert die Ansicht des Szenarios
	Herauszoomen	Verkleinert die Ansicht des Szenarios
	„Zoom Fit“	Passt die Ansicht des Szenarios an
	Löschen	Löscht ausgewählte Elemente aus dem Szenario
	Einstellungen	Öffnet das Einstellungsmenü von Maverig
	Automatisches Ausrichten	Startet die automatische visuelle Anpassung des Szenarios. S. Kapitel Bedienung, Automatisches Ausrichten.

### 3. Moduspanel

Das Moduspanel enthält alle verfügbaren Komponenten von Maverig. Diese Komponenten können zu Erstellung eines Szenarios genutzt werden.

<b>Modus</b>		<b>Kurzbeschreibung</b>
	Selektions-Modus	Aktivieren des Selektions-Modus.
	Shift-Modus	Aktivieren des Shift-Modus.
	Komponente hinzufügen	Startet den Wizard für das Hinzufügen von neuen Komponenten.

Netz		Kurzbeschreibung
	Referenzbus	Der Referenzbus dient als Spannungsquelle bzw. als Zugangspunkt zu einem übergeordneten Energienetz und stellt elektrische Energie für das Szenario bereit.
	Knotenpunkt	Der Knotenpunkt dient als Verbindungs- und Anschlusspunkt von Leitungen, Produzenten, Konsumenten und Prosumenten.
	Transformator	Der Transformator wandelt elektrische Energie aus einer Spannungsebene in eine andere um.
	Leitung	Die Leitung dient als Verbindungsstück zwischen Knotenpunkten oder als Verbindung zwischen Produzenten, Verbrauchern, Prosumenten und Knotenpunkten.
<b>Erzeuger</b>		
	Photovoltaik	Photovoltaikanlage zur Erzeugung elektrischer Energie.
	Windenergieanlage	Windenergieanlage zur Erzeugung elektrischer Energie.
	Wärme-Kraft-Kopplung (Bio-Gas-Anlage)	Wärme-Kraft-Kopplung (Bio-Gas-Anlage) zur Erzeugung elektrischer und thermischer Energie.
<b>Verbraucher</b>		
	Haushalt	Verbraucher von Energie. Der Verbrauch ist hierbei abhängig von der Anzahl der Wohneinheiten und der Personenanzahl je Wohneinheit.
<b>Prosument</b>		
	Elektrofahrzeug	Elektrofahrzeug dient zur Speicherung sowie zum Verbrauch elektrischer Energie

#### 4. Eigenschaftenpanel

Je nach ausgewählter Komponente stellt das Eigenschaftenpanel verschiedene Anpassungsoptionen bereit. Bei Auswahl mehrerer gleichartiger Komponenten ist auch eine parallele Änderung der Eigenschaften möglich.

Komponente	Eigenschaften
Referenzbus	Basisspannungshöhe in kV
Knoten	Basisspannungshöhe in kV
Transformator	Trafo-Typ Spannungsabgriffe Online-Modus
Leitung	Leitungstyp Länge in km Online-Modus
Photovoltaik	CSV-Datei
Windenergieanlage	CSV-Datei
Wärme-Kraft-Kopplung	CSV-Datei
Haushalt	CSV-Datei
Elektrofahrzeug	CSV-Datei

5. *Statusbar*

Die Statusbar stellt verschiedene Informationen bereit und hilft Ihnen im Umgang mit Maverig. Sie stellt Informationen in 3 verschiedenen Kategorien dar. Blaue Statusmeldungen teilen mit, in welchem Modus Sie sich befinden, Grüne Statusmitteilungen geben Rückmeldung über gültige Verbindungen zwischen Komponenten und Rote Statusmeldungen weisen auf ungültige Verbindungen hin. Siehe hierzu auch Kapitel Fehlermeldungen.

6. *Szenariopanel*

Im Szenariopanel können Sie mittels der Komponenten Ihr gewünschtes Szenario erstellen. Die Szenariogröße verhält sich hierbei dynamisch und vergrößert bzw. verkleinert sich automatisch je nach Notwendigkeit.

7. *Konsole*

Die Konsole stellt Informationen über den Ablauf der Simulation bereit. So werden beispielsweise Informationen über das Starten der benötigten Simulatoren sowie dem Simulationsfortschritt ausgegeben.

Sobald die Simulation gestartet wird, passt sich die GUI entsprechend an. Wichtige Bedienelemente wie die Menüleiste, Toolbar, Statusbar und Konsole bleiben erhalten und stehen weiterhin zur Verfügung. Das Komponenten- und Eigenschaftenpanel werden während der Simulation ausgeblendet. Im Szenariopanel

(2) ist der Verlauf der Simulation zu verfolgen. Bei Klick auf eine beliebige Komponente werden die entsprechenden Parameter im Attributpanel (1) angezeigt. Der zeitliche Verlauf ist dem Fortschrittsbalken (3) zu entnehmen.

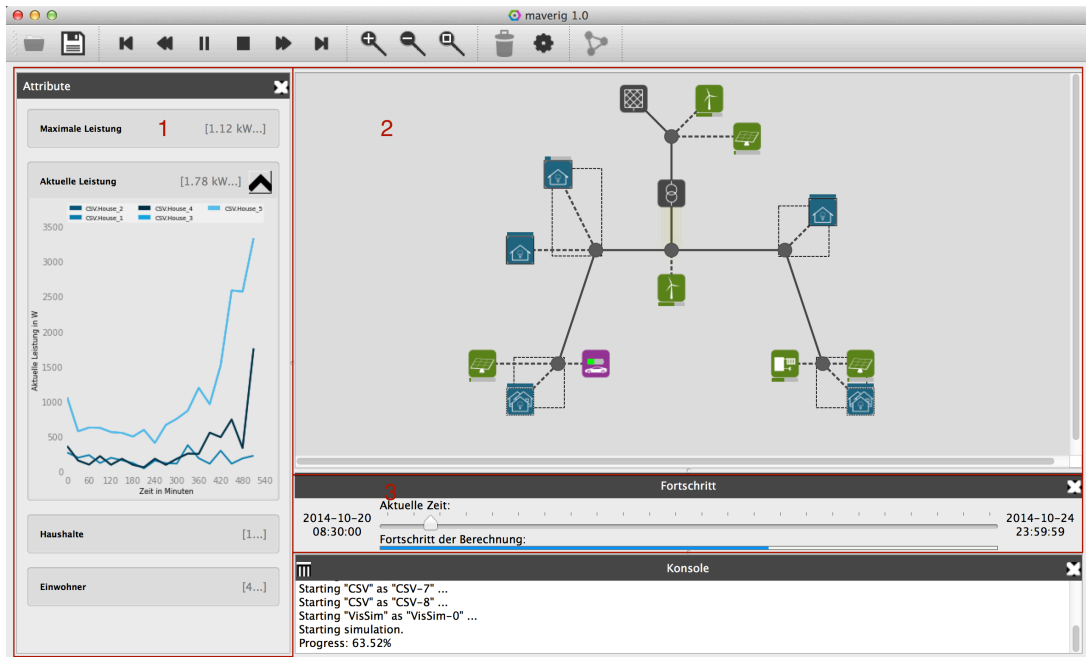


Abbildung A.4.: Übersicht GUI Maverig Simulation

1. *Attributpanel*

Das Attributpanel ist eines der zentralen Elemente während der Simulation. Hier werden alle Attribute der ausgewählten Komponente(n) dargestellt. Variable Attribute die sich während der Simulation ändern werden visuell über entsprechende Graphen dargestellt. Je nach Wunsch können hier Graphen aus- bzw. eingeblendet werden. Bei Doppelklick auf eine Komponente im Szenariopanel werden alle Komponenten gleicher Art ausgewählt. Hierbei werden die variablen Attribute in Graphen zusammengefasst wenn sie die gleiche physikalische Einheit besitzen. Somit ist es komfortabel möglich beliebige Attribute zu vergleichen. Dies ist natürlich auch dann möglich, wenn unterschiedliche Elemente im Szenariopanel markiert werden.

2. *Szenariopanel*

Sobald die Simulation gestartet wird, befindet sich das Szenariopanel im Ansichtsmodus und Veränderungen des Szenarios sind nicht mehr möglich. Einzelne Komponenten können mit einem Linksklick ausgewählt werden. Bei der Auswahl mehrerer Komponenten besteht die Möglichkeit diese mittels gedrückter STRG-Taste und dem Anwählen der Komponenten via Linksklick

zu markieren. Alternativ kann auch mittels gedrückter linker Maustaste ein “Rahmen” um die gewünschten Elemente gezogen werden.

Der jeweilige Status der einzelnen Komponenten wird im Szenariopanel über verschiedene Darstellungsformen visualisiert. Diese können in den Einstellungen von Maverig personalisiert werden. Folgende Optionen stehen zu Auswahl:

Kategorie	Optionen
Netzkomponenten	- Schatteneffekt - Farbeffekt
Produzenten, Konsumenten und Prosumenten	- Balkeneffekt - Schatteneffekt - Transparenzeffekt

### 3. Fortschrittsbalken

Dem Fortschrittsbalken können Sie auf der linken Seite den Startzeitpunkt (1) und auf der rechten Seite den Endzeitpunkt der Simulation (2) entnehmen. Der Scrollbalken (3) bewegt sich entsprechend dem definierten Geschwindigkeitsfaktor, der in den Simulationsoptionen eingestellt werden kann und gibt die aktuelle Position der Simulation an. Unterhalb des Scrollbalkens befindet sich der bereits berechnete Simulationsfortschritt (4). Mittels des Scrollbalkens können Sie an jeden beliebigen, bereits berechneten Zeitpunkt springen.

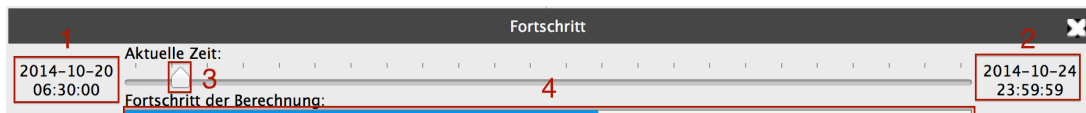


Abbildung A.5.: Fortschrittsbalken

### e. Bedienung

Das Kapitel Bedienung befasst sich im Detail mit der genaueren Benutzung von Maverig. Hierbei wird anhand von verschiedenen Arbeitsprozessen die Bedienung der Software erläutert. Maverig umfasst vier verschiedene Modi, den Selektions-Modus, den Shift-Modus, den Komponenten-Modus und den Simulations-Modus. Im Selektions-Modus können sämtliche Elemente im Szenario ausgewählt und bewegt werden. Sobald eine Komponente im Moduspanel ausgewählt wird, wechselt der Modus automatisch in den Komponenten Modus. Dieser Modus ermöglicht das Platzieren von Elementen im Szenario. Zusätzlich kann der Shift-Modus über das entsprechende Icon in der Toolbar oder der Menüleiste ausgewählt werden um das Szenario zu betrachten bzw. sich durch das Szenario zu

bewegen. In diesem Modus können keine Elemente platziert, bewegt oder ausgewählt werden. Sobald eine Simulation gestartet wird, wechselt Maverig in den Simulations-Modus.

#### 1. Elemente platzieren

Um ein Element im Szenario zu platzieren, wählen wir die gewünschte Komponente mit einem Links-Klick im Moduspanel aus. Die Auswahl wird uns visuell durch eine Vergrößerung des angewählten Icons dargestellt. Maverig wechselt bei der Auswahl einer Komponente automatisch in den Komponenten-Modus, welcher das Platzieren von Elementen ermöglicht. Dies wird uns zusätzlich durch eine Meldung der Statusbar mitgeteilt. Anschließend klicken wir im Szenario an die gewünschte Stelle, an der unsere Komponente erstellt werden soll. Alternativ können wir die Elemente auch via Drag& Drop platzieren. Einige Komponenten erfordern nach dem Platzieren im Szenario das Direkte erstellen einer Leitung. Bei diesen Komponenten wird nach der Platzierung automatisch eine Leitung erstellt, die mittels Maus an die gewünschte Position gebracht werden kann. Mit einem weiteren Links-Klick wird die Leitung im Szenario gezeichnet. Ein direktes Verbinden mit anderen Komponenten ist ebenfalls möglich.

#### 2. Position von Elementen ändern

Im Komponenten-Modus oder dem Selektions-Modus lässt sich die Position einer oder mehrere Elemente im Szenariopanel verändern. Hierzu wählen Sie das gewünschte Element mit einem Links-Klick aus und verschieben es mittels Drag& Drop an die gewünschte Stelle. Bei Auswahl eines Elements mit einem Doppelklick, werden alle Elemente dieser Art ausgewählt. So können mehrere Elemente auf einmal verschoben werden. Zusätzlich ist eine Mehrfachauswahl im Selektions-Modus möglich. Durch halten der linken Maustaste kann ein „Rahmen“ zur Auswahl um die Elemente gezogen werden. Die ausgewählten Elemente können dann ebenfalls via Drag& Drop im Szenario bewegt werden.

#### 3. Eigenschaften von Elementen ändern

Um die Eigenschaften eines Elements zu ändern, muss dieses im Szenariopanel ausgewählt werden. Im Eigenschaftenpanel werden dann die entsprechenden Optionsmöglichkeiten des ausgewählten Elements aufgeführt. Zusätzlich ist es möglich die Optionen von mehreren gleichen Elementen des gleichen Komponententyps zu ändern.

#### 4. Verbinden von Elementen

Zur Verbindungsherstellung zwischen den Elementen bietet Maverig zwei verschiedene Möglichkeiten. Zum einen über die Komponente *Leitung*, zum anderen automatisch nach dem Platzieren einer Komponente. Zur Erstellung einer Leitung zwischen 2 Elementen wählen Sie das entsprechende Symbol aus



dem Moduspanel und anschließend den gewünschten Startpunkt im Szenario mit einem weiteren Linksklick. Nun führen Sie die Verbindung zu dem gewünschten Endpunkt. Maverig teilt Ihnen hierbei über die Statusbar mit, ob diese Verbindung technisch möglich ist. Abschließend können Sie mit einem weiteren Linksklick die Leitung bzw. Verbindung erstellen.

5. Entfernen von Elementen  
Zur Entfernung von Elementen wählen Sie diese mit einem Links-Klick im Szenario aus. Anschließend können sie bspw. über die Entfernen-Taste Ihrer Tastatur oder durch das Betätigen des Löschen-Icons in der Toolbar das gewählte Element entfernen. Um mehrere Elemente gleichzeitig aus dem Szenario zu entfernen markieren Sie diese vorher.
6. Kopieren/ Ausschneiden/ Einfügen von Elementen  
Zum Kopieren bzw. Ausschneiden von Elementen markieren Sie es im Szenario. Anschließend nutzen Sie den entsprechenden Shortcut auf der Tastatur oder den Punkt aus der Menüleiste. Beim Einfügen der Elemente aus der Zwischenablage verfahren Sie genauso. Siehe auch Abschnitt Shortcuts.
7. Zoom und Zoom Fit  
Ist Ihr erstelltes Szenario für das Szenariopanel zu groß, ist es möglich mittels Zoom die Ansicht zu vergrößern bzw. zu verkleinern. Dazu können Sie das Mousrad Ihrer Maus oder die entsprechenden Buttons in der Toolbar bzw. der Menüleiste nutzen. Maverig bietet mit der Zoom Fit Funktion die zusätzliche Möglichkeit, die Zoomstufe automatisch der Größe des Szenarios anzupassen. So haben Sie Ihr gesamtes Szenario im Blick. Zoom Fit führen Sie über den entsprechenden Button in der Toolbar bzw. der Menüleiste aus.
8. Automatische Darstellungsoptimierung des Szenarios  
Maverig bietet mit der Funktion “Automatisches Ausrichten” die Möglichkeit die Elemente Ihres Szenarios automatisch Anzuordnen um eine bessere Übersichtlichkeit zu ermöglichen. Über den entsprechenden Button in der Toolbar oder der Menüleiste können Sie die automatische Darstellungsoptimierung starten. Anschließend wird der Optimierungsprozess visuell im Szenario dargestellt, damit Sie ihn verfolgen können.
9. Simulationszeit einstellen  
Bevor Sie die Simulation starten ist es möglich den Start- und Endzeitpunkt sowie den Geschwindigkeitsfaktor und das Intervall der Simulation festzulegen. Dies erfolgt über den Punkt *Zeit einstellen* in der Menüleiste. Maverig speichert Ihre gewählte Einstellung bis zu weiteren Änderungen. Über den Geschwindigkeitsfaktor können Sie die Wiedergabegeschwindigkeit der Simulation definieren.
10. Simulation starten

Wenn Sie die Erstellung Ihres Szenarios abgeschlossen haben oder einen Zwischenstand vorab Simulieren möchten, können Sie die Simulation starten. Dies erfolgt über den Start Button in der Toolbar oder den entsprechenden Punkt in der Menüleiste.

11. Attribute von Elementen während der Simulation anzeigen  
Während der Simulation steht Ihnen standardmäßig das Attributepanel auf der linken Seite zur Verfügung. Bei Auswahl eines Elements im Szenariopanel über einen Linksklick werden hier entsprechende Eigenschaften des ausgewählten Elements angezeigt. Handelt es sich um variable Attribute, die sich während der der Simulation ändern, werden diese visuell über einen Graphen dargestellt.
12. Simulationsgeschwindigkeit erhöhen/verringern  
Über den Button *Geschwindigkeit Erhöhen* in der Toolbar bzw. der Menüleiste können Sie durch mehrmaliges Klicken die Simulationsgeschwindigkeit erhöhen. Dadurch wird die Simulation beschleunigt wiedergegeben, allerdings nur soweit wie bereits Daten aus der Simulationsberechnung vorliegen. Alternativ können Sie auch über den Scrollbalken an einen bereits errechneten Punkt in der Simulation springen. Natürlich können Sie auch analog die Simulationsgeschwindigkeit verringern.
13. Simulation stoppen  
Über den Button *Stoppen* in der Toolbar bzw. der Menüleiste können Sie die Simulation vorzeitig beenden. Damit gelangen Sie zurück in den Kompositionsmodus von Maverig, in dem Sie Anpassungen an Ihrem Szenario vornehmen können.

#### f. **Komponentenassistent**

Der Komponentenassistent ermöglicht das geführte Erstellen von neuen Komponenten in Maverig. Er führt Sie Schritt für Schritt durch den Erstellungsprozess. Hier soll kurz die Bedienung des Komponentenassistenten aufgezeigt werden. Sie starten den Erstellungsprozess mit einem Klick auf das entsprechende Symbol im Moduspanel. Anschließend startet der Assistent in einem neuem Fenster.

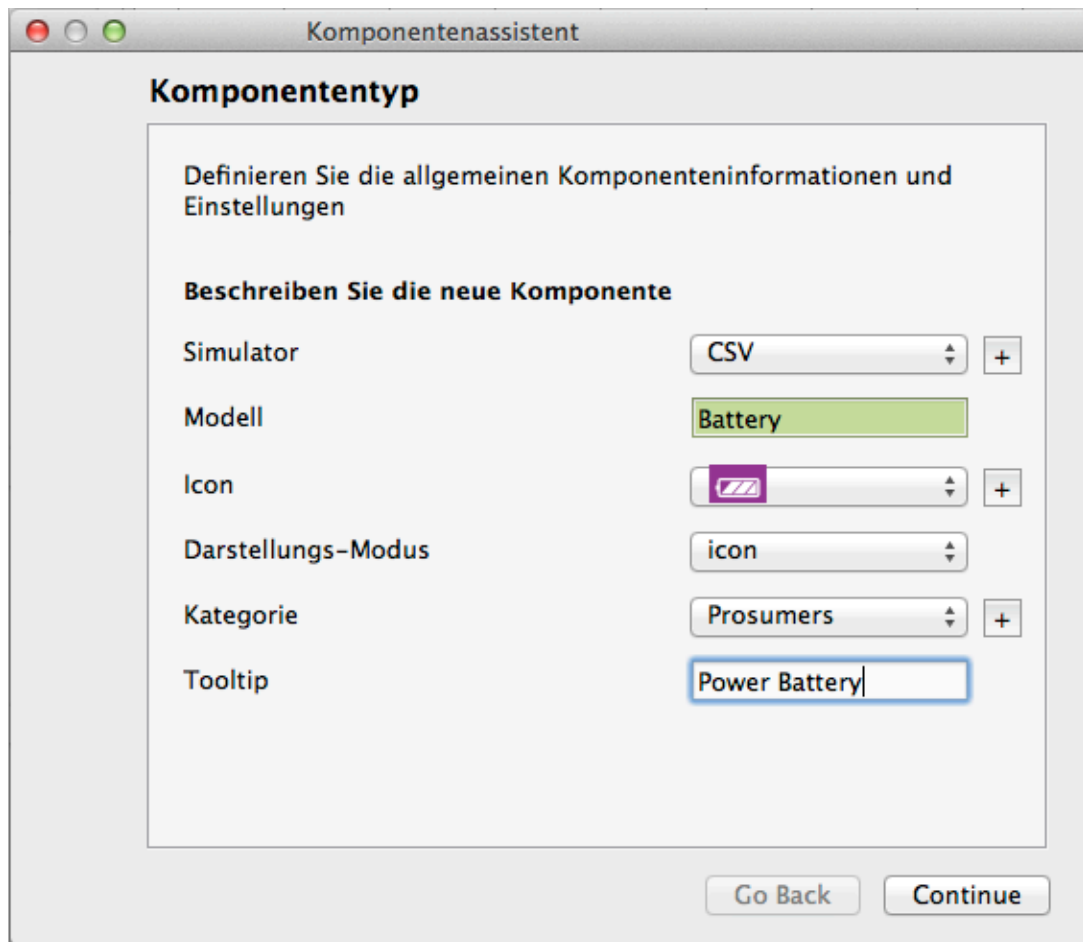


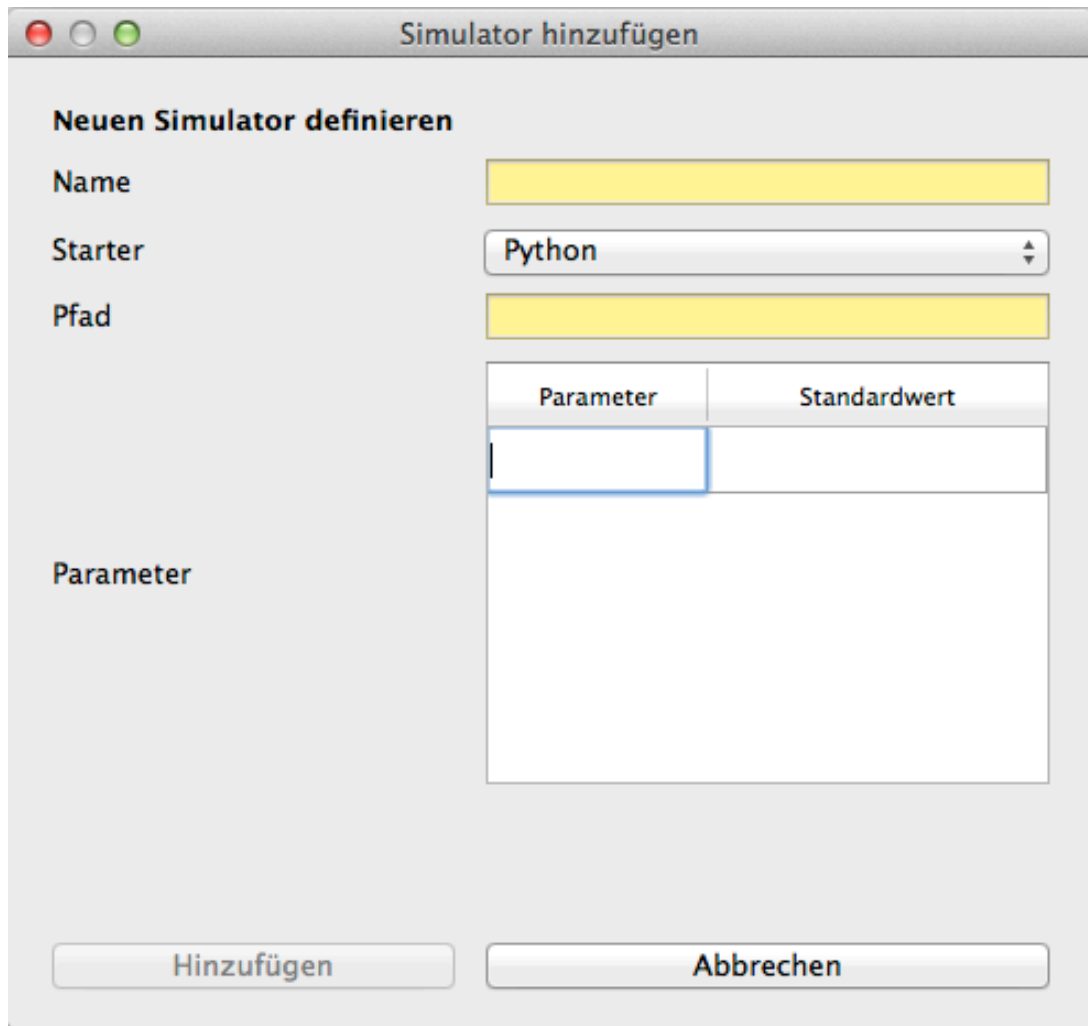
Abbildung A.6.: Komponentenassistent

Hier können Sie zunächst die Art des Simulators wählen, bspw. PyPower oder CSV. Alternativ können Sie, über das entsprechende +-Symbol, einen neuen Simulator hinzufügen. Des Weiteren muss ein Modellname vergeben sowie ein Icon für die neue Komponente ausgewählt werden. Standardmäßig stehen eine Reihe von verschiedenen Symbolen zur Verfügung. Über das +-Symbol können Sie eigene Icons hinzufügen.

Bei der Wahl des Darstellungsmodus stehen die Optionen *line*, *line-icon-line*, *icon* und *node* zur Auswahl. Je nach Anwendungsart der neuen Komponenten wählen Sie hier die entsprechende Option. Abschließend können Sie die Kategorie definieren, in der die Komponente später im Modepanel einsortiert wird. Auch hier ist es möglich eine neue Kategorie zu erstellen. Optional ist das Festlegen eines Tooltips möglich. Für den nächsten Schritt klicken Sie auf Weiter.

### Neuen Simulator hinzufügen

Fügen Sie einen neuen Simulator hinzu, öffnet sich ein neues Fenster. Hier müssen Sie einige Felder für den neuen Simulator ausfüllen.



**Neuen Simulator definieren**

Name

Starter

Pfad

Parameter	Standardwert
<input type="text"/>	<input type="text"/>

Parameter

Abbildung A.7.: Hinzufügen eines neuen Simulators

Benötigte Angaben für den neuen Simulator erkennen Sie an den gelb hinterlegten Feldern, die ausgefüllt werden müssen. So werden z.B. der Simulatorname, der Starter und der Pfad an dem sich der neue Simulator befindet benötigt. Zudem können Sie noch Parameter definieren. Anschließend können Sie mit einem Klick auf *Hinzufügen* den Simulator der neuen Komponenten hinzufügen. Sie gelangen danach zurück in das Startfenster des Assistenten.

### Parameter und Attribute festlegen

Im nächsten Schritt des Komponentenassistenten legen Sie Parameter, Attribute und ihre Beschreibungen fest.

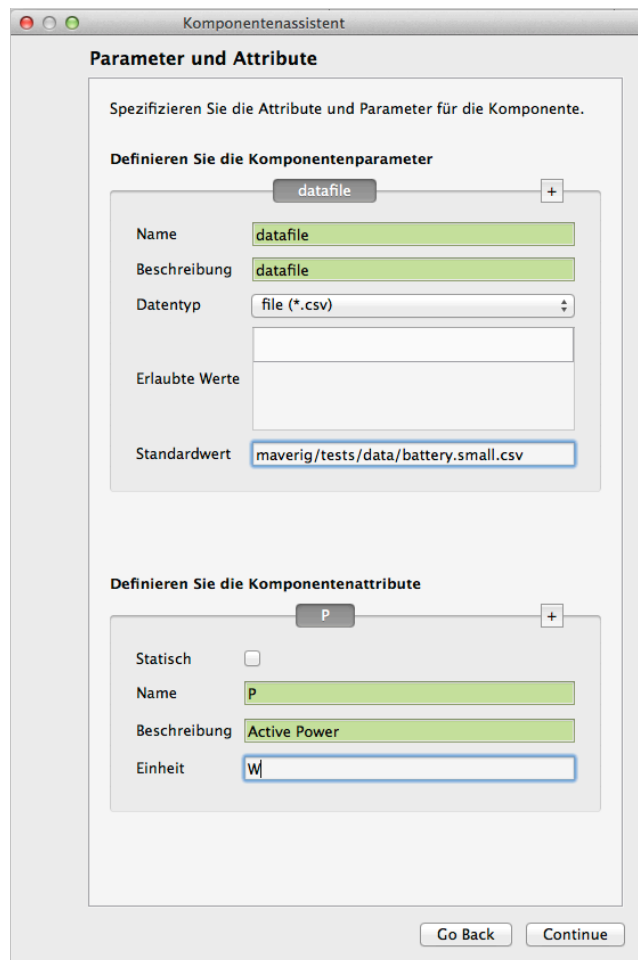


Abbildung A.8.: Parameter und Attribute festlegen

Parameter und Attribute müssen mit einem Namen und einer Beschreibung versehen werden. Zusätzlich müssen Sie bei Parametern den Datentyp angeben und können für diesen einen Standardwert definieren. Auch hier können Sie über das +-Symbol beliebig viele Parameter oder Attribute hinzufügen. Bei Attributen muss zusätzlich eine Einheit angegeben werden. Wählen Sie die Option *statisch*, wird dieser Parameter während der Simulation nicht als Graph dargestellt, da davon ausgegangen wird, dass sich der Attributwert im Simulationsverlauf nicht ändert. Abschließend klicken Sie erneut auf weiter und schließen den Komponentenassistenten ab.



Nach Abschluss des Komponentenassistenten finden Sie Ihre neu erstellte Komponente in der ausgewählten Kategorie im Moduspanel wieder. Sie können Sie nun in Ihrem Szenario verwenden.

**g. Fehlermeldungen und Problemlösungen**

<b>Fehlermeldung</b>	<b>Beschreibung</b>
Komponente konnte nicht erstellt werden - ungültige Verbindung	Sie haben versucht ein neues Element direkt über ein bereits bestehendes zu platzieren. Bitte platzieren Sie das Element in einem freien Bereich des Szenarios.
Keine Elemente zum Verbinden	Es wurden keine Elemente zum Verbinden gefunden.
Leitungslänge muss länger als 0 km sein	Sie haben eine Leitungslänge von 0 km definiert.
Ungültige Verbindung	Sie haben versucht eine nicht valide Verbindung zu erstellen.
Startzeit muss kleiner als Endzeit sein	Die Simulationsstartzeit muss vor der Simulationsendzeit liegen.
Intervallgröße muss kleiner als die Simulationszeit und größer als 0 sein	Sie haben eine Intervallgröße gewählt, die größer als die Simulationszeit ist.
Die Länge der ausgewählten Leitung ist zu lang	Aufgrund der Leitungslänge und der zu übertragenen elektrischen Energie erfolgte ein Simulationsabbruch. Bitte ändern Sie die Leitungslänge bzw. entsprechende Parameter (Bspw. Leitungstyp).

**h. Shortcuts**

Steuerungstaste Windows/Linux: Strg  
 Steuerungstaste Mac: cmd

<b>Shortcut</b>	<b>Funktion</b>
Steuerungstaste + ,	Einstellungen
Steuerungstaste + H	Maverig ausblenden
Steuerungstaste + N	Neue Datei
Steuerungstaste + O	Datei Öffnen
Steuerungstaste + S	Datei Speichern
Steuerungstaste + Shift + S	Datei Speichern unter
Steuerungstaste + Q	Maverig Schließen
Steuerungstaste + Z	Bearbeiten Rückgängig
Steuerungstaste + Y	Bearbeiten Wiederherstellen
Steuerungstaste + X	Ausschneiden
Steuerungstaste + Z	Kopieren
Steuerungstaste + V	Einfügen
Steuerungstaste + A	Alles Markieren
F1	Hilfe
F4	Automatisches Ausrichten
F5	Simulation Ausführen
F6	Simulation Stoppen
F7	Simulation Pause
F8	Zum Start der Simulation springen
F9	Simulationsgeschwindigkeit reduzieren
F10	Simulationsgeschwindigkeit erhöhen
F11	Zum Ende der Simulation springen
Steuerungstaste + T	Simulationszeit einstellen
Steuerungstaste + G	Springe zu Simulationszeitpunkt
Steuerungstaste + Alt + S	Selektions-Modus
Steuerungstaste + Alt + R	Raster aktivieren/deaktivieren
Steuerungstaste + +	Vergrößern
Steuerungstaste + -	Verkleinern
Steuerungstaste + .	Szenario anpassen
Steuerungstaste + 1	Moduspanel ein-/ausblenden
Steuerungstaste + 2	Eigenschaftenpanel ein-/ausblenden
Steuerungstaste + 3	Konsole ein-/ausblenden
Steuerungstaste + 4	Statusleiste ein-/ausblenden
Steuerungstaste + 5	Fortschrittsleiste ein-/ausblenden
Steuerungstaste + 6	Attributpanel ein-/ausblenden



## A.2. User Manual

### a. Introduction

Maverig is a graphical User Interface for creation and visualization of Smart-Grid simulations. Maverig is divided into the Composition Mode and the Simulation Mode. In Composition Mode a Smart-Grid scenario can be created and afterwards the user can run a simulation of this scenario in Simulation Mode while observing significant parameters. For this purpose the Mosaik simulators are used by Maverig.

### b. Setup User Manual

This User Manual gives a general overview of the functionality and usability of Maverig. Therefore the User Manual is divided into two parts. First there is an explanation how to use the Composition Mode to create a scenario. Second there is a description how to use the Simulation Mode.

### c. Installation

Maverig supports the operation systems Linux, OSX and Windows. In addition there is only a support for Python 3.4 or higher. For a Maverig installation the paketmanager „pip“ is required. Python 3.4 already includes pip. The command to install Maverig with the pip paketmanager is: “pip install maverig”

### d. Setup GUI

This chapter contains the setup of the Graphical User Interface (GUI). Below there is a detailed explanation for all GUI components and their functionality. Basically the GUI setup for Composition Mode and Simulation Mode is very similar. In Composition Mode the GUI is divided into a Menubar (1), a Toolbar (2), a Modepanel (3), a Propertypanel (4), a Statusbar (5), a Scenariopanel (6) and a Console (7).



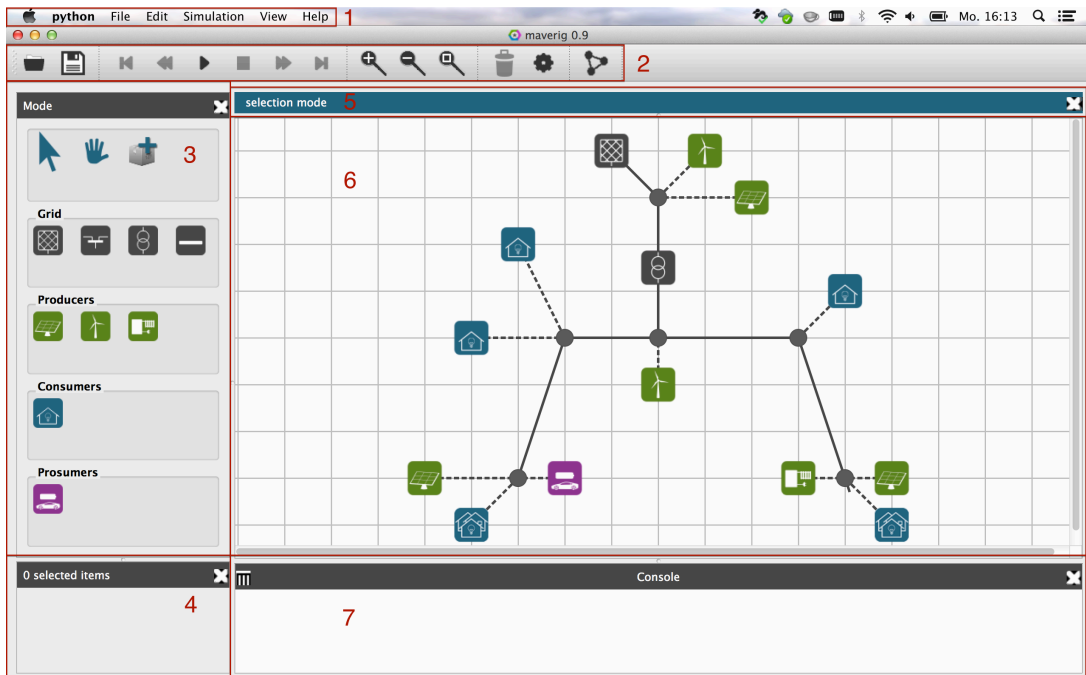


Abbildung A.9.: Overview GUI Maverig Composition

1. *Menubar*

Via the Menubar there are different features and settings for Maverig available. Many features in the Menubar are also in the Toolbar.

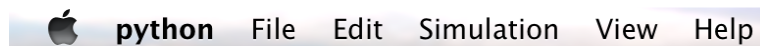


Abbildung A.10.: Maverig Menubar















File	New	Creates an empty scenario
	Open	Opens an existing Maverig scenario
	Save	Saves the scenario
	Save as	Saves the scenario in chosen path
	Preferences	General settings and specific simulation settings in Maverig
	Quit	Closes Maverig
Edit	Undo	Undoes the last step performed in the scenario
	Redo	Reverts the last step performed in the scenario
	Auto Layout	Arranges elements in Scenariopanel clearly
	Cut	Cut-out selected elements and paste into clipboard
	Copy	Copy selected elements and paste into clipboard
	Paste	Inserts elements from clipboard into scenario
	Delete	Deletes selected elements from scenario
	Select All	Selects all elements from scenario
Simulation	Run	Start the simulation
	Stop	Stop the simulation
	Pause	Pause the simulation
	Back to start	Skip to starting time of simulation
	Reduce Speed	Reduce playback speed of simulation
	Increase Speed	Increase playback speed of simulation
	Forward to end	Skip to end time of simulation
	Set time	Adjustment of simulation start and end time
	Go to	Skip to declared time of simulation
View	Shift Mode	Enable the Shift Mode
	Selection Mode	Enable the Selection mode
	Raster	Enable or Disable automatic docking of elements at Raster
	Zoom In	Increase the view of the scenario
	Zoom Out	Decrease the view of the scenario
	Zoom Fit	Adjust the view of the scenario automatically
	Fading out GUI components	Fade GUI components in or out.
Help	Maverig Help	Opens Maverig Help
	About Maverig	Informationen about Maverig e.g. involved developers

## 2. *Toolbar*

Via buttons in the Toolbar different features in Maverig are available. All features in the Toolbar are part of the Menubar.






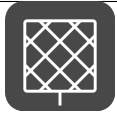








Abbildung A.11.: Maverig Toolbar

	Open	Open an existing Maverig scenario
	Save	Save the scenario
	Back to start	Skip to start time of simulation
	Reduce Speed	Reduce playback speed of simulation
	Run	Start the simulation
	Stop	Stop the simulation
	Increase Speed	Increase playback speed of simulation
	Forward to end	Skip to end time of simulation
	Zoom In	Increase the view of the scenario
	Zoom Out	Decrease the view of the scenario
	Zoom Fit	Adjust the view of the scenario automatically
	Delete	Deletes selected elements from scenario
	Settings	General settings and specific simulation settings in Maverig
	Auto Layout	Arranges elements in Scenariopanel clearly q.v. chapter Operation, Auto Layout

### 3. Modepanel

The Modepanel contains every usable component in Maverig. These components can be used to create a scenario.

	<b>Mode</b>	<b>Description</b>
	Selection Mode	Enable the Selection Mode
	Shift Mode	Enable the Shift Mode
	Add Component	Start the wizard to add new components

<b>Grid</b>		<b>Description</b>
	Reference Bus	The Reference Bus is a voltage source and access point to a superior power grid and provides electric power for the scenario.
	PQBus Node	The PQBus is a connection point for lines, producers, consumers and prosumers.
	Transformer	The Transformer converts electric power from one voltage level into another voltage level.
	Line	The Line is a link between PqBus elements or a connection between producers, consumers, prosumers and PQBus elements.
<b>Producer</b>		
	Photovoltaic	A Photovoltaic facility to produce electric power.
	Wind Energy Conversion System	A Wind Energy Conversion System to produce electric power.
	Cogeneration of Heat and Power	Using Cogeneration of Heat and Power to produce electric power.
<b>Consumer</b>		
	Household	Consumer of electric power. The Consumption depends on the quantity of apartments and the number of residents per apartment.
<b>Prosumer</b>		
	Electronic Vehicle	An Electronic Vehicle produces and consumes electric power.

#### 4. Propertypanel

Depending on the selected component the Propertypanel provides a set of adjustable parameters for this element. If several congeneric components are selected, the properties can be adjusted simultaneously.

Component	Property
Reference Bus	Basic Stress Level in kV
PQBus Node	Basic Stress Level in kV
Transformer	Transformer Type Voltage Tap-Off Online Mode
Line	Length in km Online Mode
Photovoltaic	CSV-File
Wind Energy Conversion System	CSV-File
Cogeneration of Heat and Power	CSV-File
Household	CSV-File
Electronic Vehicle	CSV-File

#### 5. *Statusbar*

The Statusbar provides several informations and supports the user of Maverig. Informations are divided in three categories. Blue status messages convey the current mode, green status messages give feedback about valid connections between elements and red status messages give feedback about invalid connections between elements (q.v. chapter Error messages and troubleshooting).

#### 6. *Scenariopanel*

The Scenariopanel allows the user to create a specific scenerio by using all given kinds of components. The scenario size is dynamic and increases or decreases depending on its necessity simultaneously.

#### 7. *Console*

The Console provides information about the process of the simulation. Including informations about the start of required simulators as well as the simulation progress.

As soon as the simulation starts there is a correspondent GUI change. Major operating elements as the Menubar, Toolbar, Statusbar and Console remain in the GUI. The Componentpanel and Propertypanel are disabled during simulation. The process of the simulation is displayed in the Scenariopanel (2). If any component is selected, the corresponding parameters are displayed in the Attributepanel (1). The Progressbar (3) shows the chronological process of the simulation.

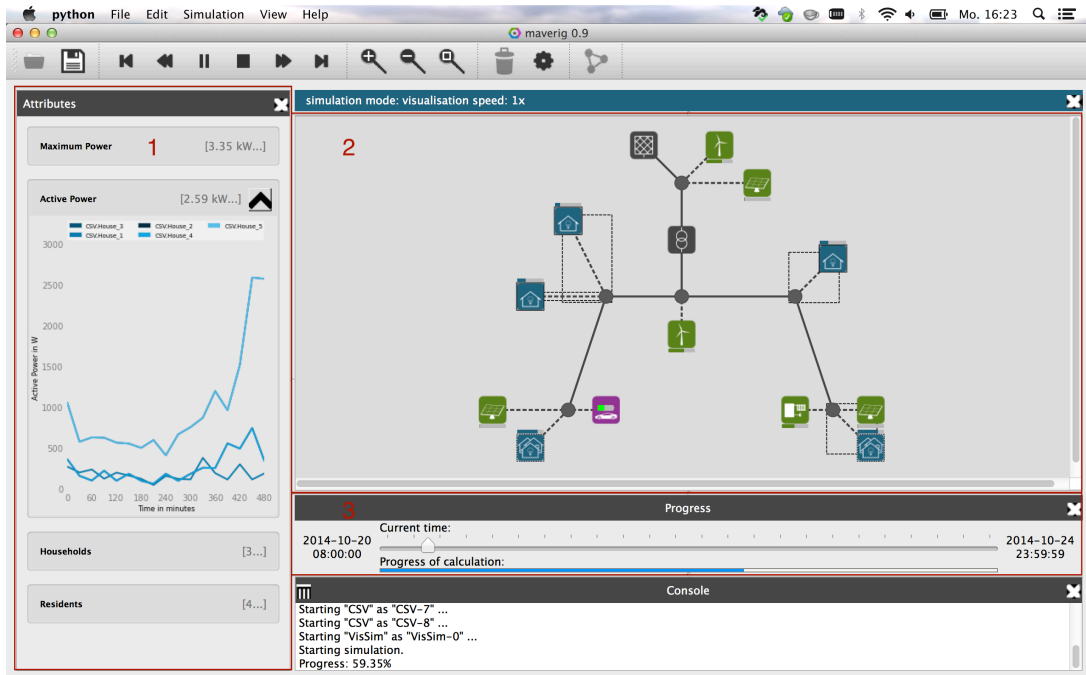


Abbildung A.12.: Overview GUI Maverig Simulation

1. *Attributepanel*

The Attributepanel is one of the essential elements during the simulation. It shows all attributes of the selected component(s). Any variable attribute, whose value changes during the simulation, is visualized in corresponding graphs. Depending on the user, graphs can be faded in or out. A double click on an element in the Scenariopanel selects any elements of these kind. Furthermore the variable attributes of these elements are summarized in one graph if they have the same measurement unit. Therefore it is possible to compare different attributes. It is also possible by selecting various elements within the Scenariopanel.

2. *Scenariopanel*

After starting the simulation the Scenariopanel is only in View Mode, thus the scenario cannot be changed temporarily. Single components can be selected by a left-hand click. Multiple components can be selected by tapping the CTRL button and selecting several components by a left-hand click. Alternatively multiple components can be selected by a tapping left-hand click and by drawing a frame around several components.

The current status of single components within the Scenariopanel is displayed in diverse types. Maverig provides special settings for the visualization of these



types. The following are available:

Category	Type
Grid Components	- Shadow Effect - Color Effect
Producer, Consumer und Prosumer	- Bar Effect - Shadow Effect - Transparency Effect

### 3. Progressbar

The Progressbar has the start time (1) of the simulation on the left side and the end time (2) of the simulation on the right side. The Scrollbar (3) moves depending on the chosen speed, which can be changed in the simulation settings. The Scrollbar also displays the current progress of the simulation. Below there is a blue bar (4), which displays the current calculated simulation progress. By using the Scrollbar the user can skip to any already calculated point of the simulation.

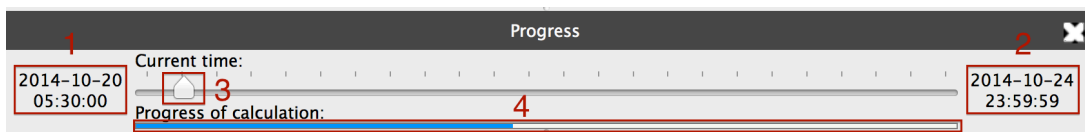


Abbildung A.13.: Progressbar

### e. Operation

The chapter operation describes how to use Maverig in detail. This will be explained on the basis of various operating processes. In general Maverig contains four different modes, the Selection Mode, the Shift Mode, the Component Mode and the Simulation Mode. In Selection Mode any element in the scenario can be selected and moved. If a component of the Modepanel is selected the mode switches into Component Mode automatically. In this mode components can be placed into the scenario. In addition the user can switch into the Shift Mode by the corresponding button in the Toolbar or Menubar. In Shift Mode the user can overview and move through the whole scenario. In this mode components cannot be placed, moved or selected. If the simulation has been started, Maverig switches into Simulation Mode simultaneously.

#### 1. Place element

To place an element into the scenario the user has to select any component in the Modepanel by a left-hand click. The selected component is displayed

through an enlarged icon in the Modepanel. If a component of the Modepanel is selected Maverig switches the mode into Component Mode automatically. In this mode components can be placed into the scenario. In addition there is also a message in the Statusbar. Afterwards the user can click on the required position, to create an instance of the selected component on this position. Alternatively the component can also be placed by Drag & Drop. For Some components there is a need to create a line, after they have been placed into the scenario. In this case Maverig creates a line automatically, which can be put to the required position using the mouse. Another left-hand click draws the line into the scenario. A direct connection to other elements is possible as well.

#### 2. Change element position

In Component Mode or in Selection Mode the position of one or several elements in the Scenariopanel can be changed. Therefore select the required elements and move them by Drag & Drop to their new position. A Selection of an element via double-click, selects any element of this type in the current scenario. So several elements can change their position at once. Secondary a multiple selection is possible in Selection Mode. The elements can be selected by a tapping left-hand click and by drawing a frame around several elements. The selected elements can be moved by Drag & Drop to their new position in the scenario.

#### 3. Change element properties

To change the properties of an element, the element needs to be selected in the Scenariopanel. In the Propertypanel the user can see the adjustable properties of the selected element. In addition it is possible to adjust properties of several elements of the same type simultaneously.

#### 4. Connect components

In Maverig there are two alternatives to establish a connection between elements. On the one hand by using the Line component and on the other hand after placing an element automatically. To establish a Line between two elements, the user has to select the component Line in the Modepanel. The first left-hand click in the scenario sets the start point. Another left-hand click in the scenario sets the end point. Maverig displays information about valid or invalid connections in the Statusbar simultaneously. If the end point creates a valid connection the Line or rather connection between both elements will be created.

#### 5. Delete elements

To delete one element or several components the user has to select the required elements in the scenario. Afterwards the element(s) can be deleted by

using the Delete-Key of the keyboard or the Delete-Icon of the Toolbar.

6. Copy/Cut/Paste elements

To copy or cut elements select the required element in the scenario. Thereafter use the corresponding keyboard shortcut or the Menubar shortcut. In the same manner elements from clipboard can be inserted into the scenario by using the paste shortcuts (q.v. chapter Shortcuts).

7. Zoom and Zoom Fit

If the created scenario is too big for the Scenariopanel the user can Zoom In or Zoom Out to adjust the view of the Scenariopanel. For that purpose the mouse wheel as well as the corresponding buttons in the Toolbar and Menubar can be used. Maverig also provides a Zoom Fit feature to adjust the zoom level to display the whole scenario in viewing range. So the user has an overview of the entire scenario. The Zoom Fit feature can be used by the corresponding button in the Toolbar or the Menubar.

8. Auto Layout

Maverig provides a feature named “Auto Layout”, which arranges elements in the Scenariopanel clearly. Auto Layout can be used by the corresponding button in the Toolbar or the Menubar. Afterwards the optimization process in the scenario will be displayed for the user.

9. Set simulation time

Before the start of the simulation there are some settings to determine. Such as the start and end time, the simulation speed and also the simulation range. This occurs in the Menubar by the item Set Time. Maverig saves the settings until the user changes them again. The simulation speed determines the playback speed of the simulation.

10. Run simulation

If the creation of the scenario is completed or the current state should be simulated, the simulation can be started by the corresponding start button in the Toolbar or Menubar.

11. Show element attributes during simulation

While the simulation runs the Componentpanel is replaced by the Attributepanel. If any element in the Scenariopanel is selected, the Attributepanel displays all attributes of this element. Any variable attribute, whose value changes during the simulation, is visualized in corresponding graphs.

12. Increase/Decrease simulation speed

With the buttons “Increase Speed” and “Decrease Speed” in the Toolbar and Menubar the playback speed of the simulation can be increased or decreased. Thereby the simulation playback increases only as far as the calculation of the



simulation proceeded. Alternatively the user can skip to any already calculated point of time by using the Scrollbar of the Progressbar.

13. Stop simulation

The simulation can be stopped by the stop button in the Toolbar or Menubar. Afterwards Maverig switches from Simulation Mode into Composition Mode. In Composition Mode the user can adjust the scenario.

f. **Component Wizard**

With the Component Wizard the user can create new components in Maverig and use them in further scenarios. There is a step by step creation process whose operation will be described below. The Component Wizard starts in a new window by clicking at the corresponding button in the Modepanel.

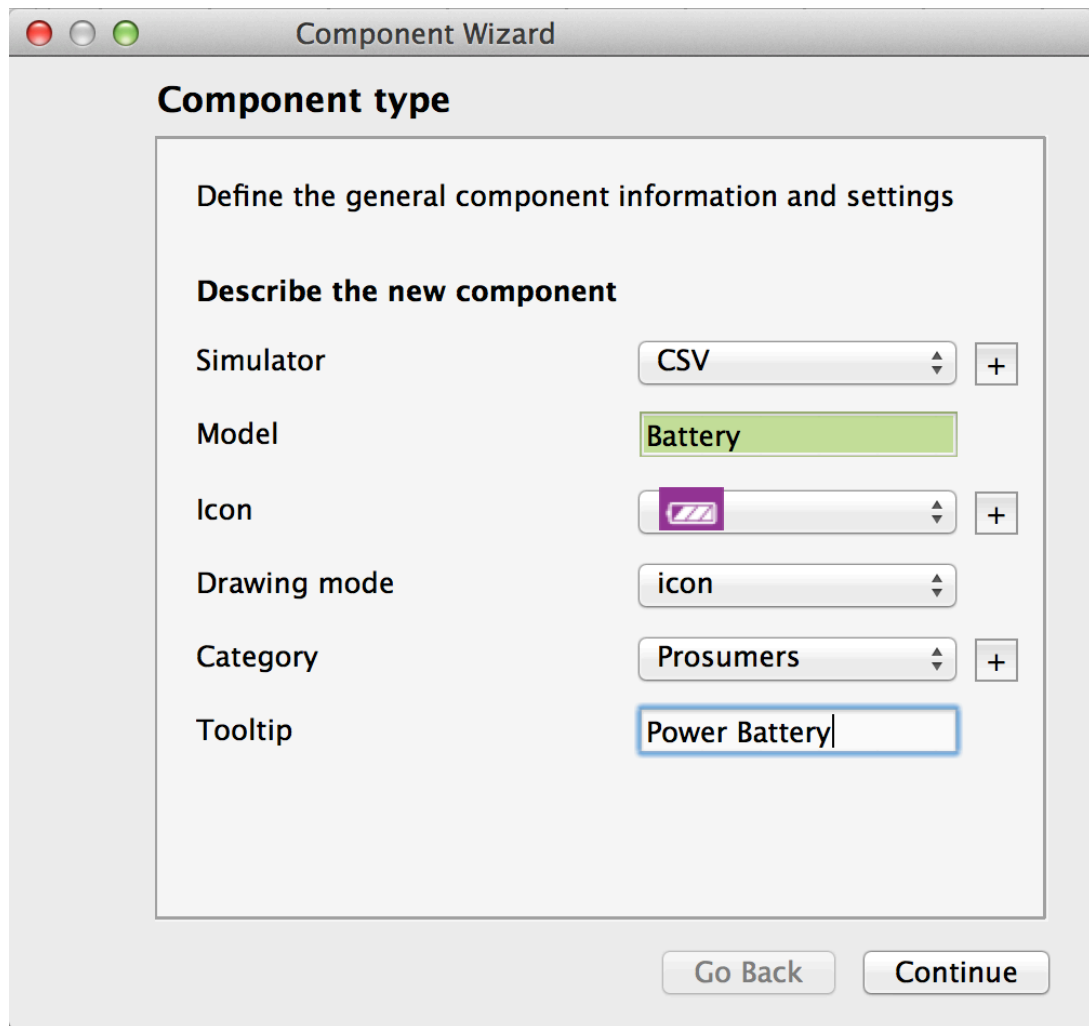


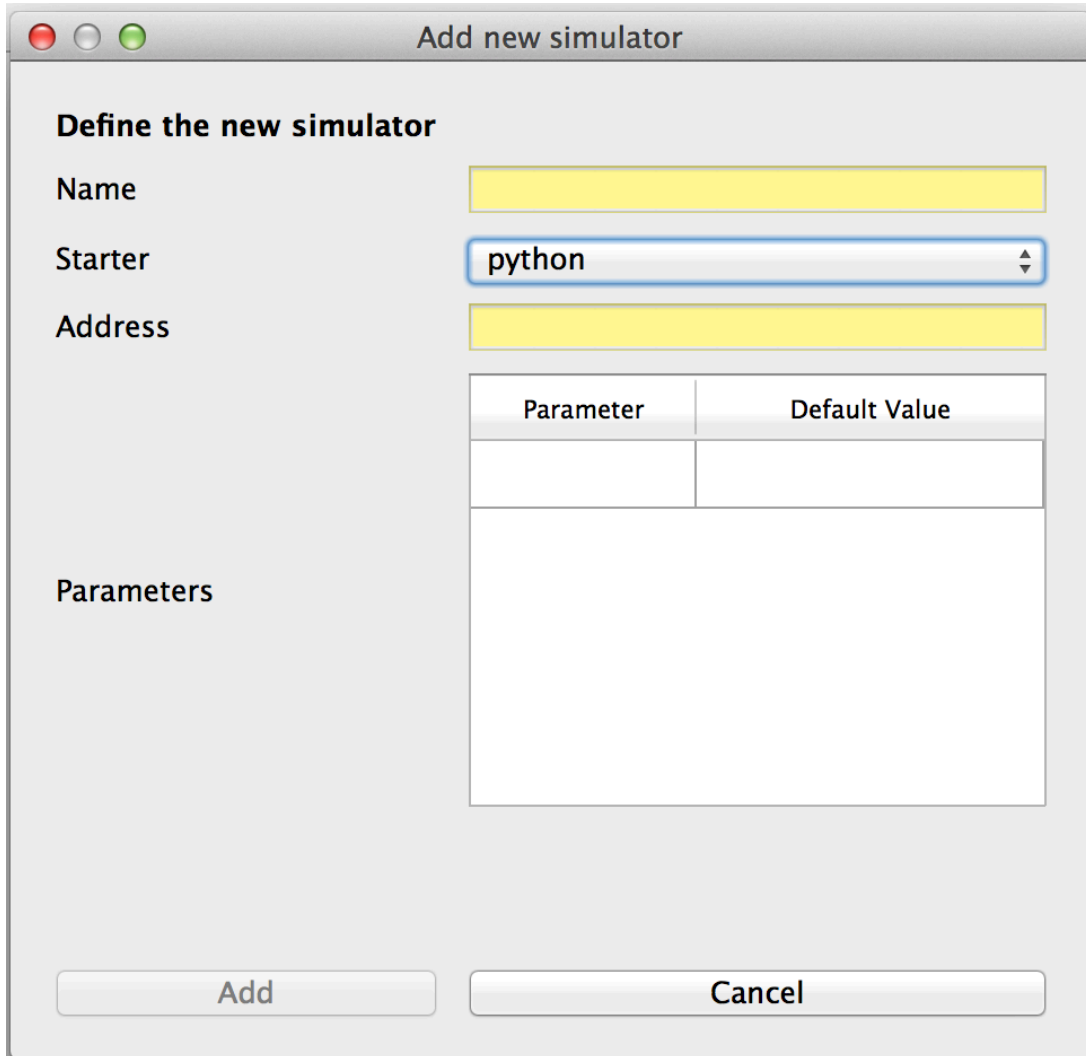
Abbildung A.14.: Component Wizard

In the beginning the user chooses the simulator type, for instance, PyPower or CSV. Alternatively the “+”-button can be used to add a new simulator. Then the new component gets a name and a representative icon. By default there are several symbols available. Here again with the “+”-button a new symbol can be added.

For the Drawing Mode there are four options as “line”, “line-icon-line”, “icon” and “node” available. The Drawing Mode depends on the type of application. Concluding the user defines the Category, in which the new component will be listed in the Modepanel in Maverig. If necessary, a new Category can be added. An optional Tooltip for the component can be assigned. The user reaches the next step by clicking on Next.

### Add new Simulator

If the user wants to add a new simulator, a new window for detailed information opens.



**Define the new simulator**

Name

Starter

Address

Parameters

Parameter	Default Value

Abbildung A.15.: Add new Simulator

The yellow areas show the required information for a new simulator as the name, the starter and the path for the new simulator. Furthermore parameters and their default values can be defined. A click on “Add”, adds the new simulator to the new component and brings the user back to the previous window.

### Determine Parameters and Attributes

In the next step of the Component Wizard parameters, attributes and their description will be determined.

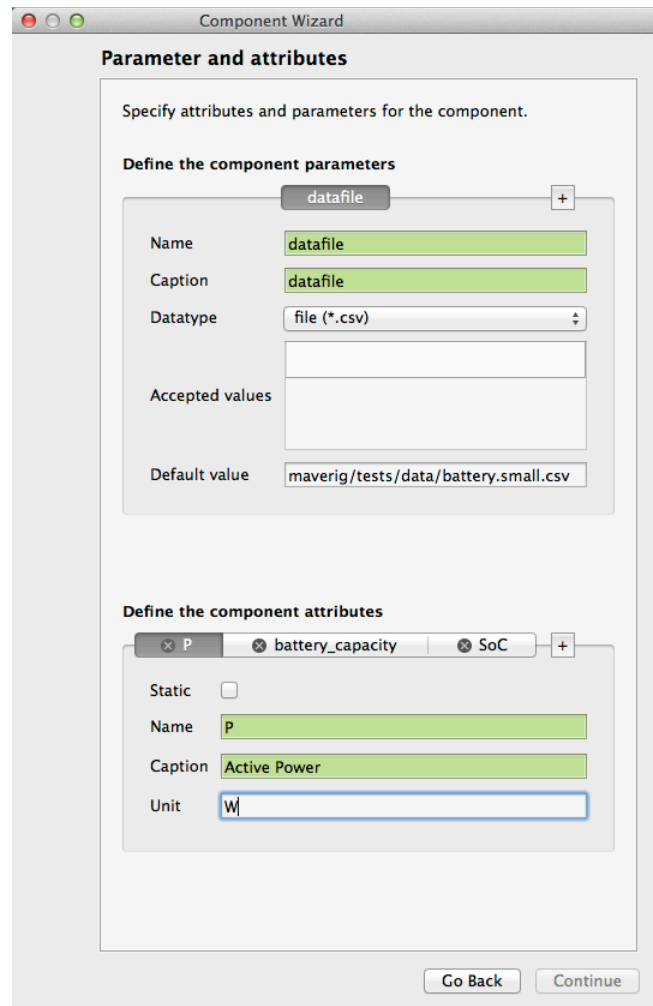


Abbildung A.16.: Determine Parameters and Attributes

Parameters and attributes need a name and a description. Additional for parameters a datatype with a default value has to be specified. For attributes a declaration of a unit is required. If the option “static” is enabled, this parameter is not displayed in a graph during simulation. Static in this context means unchanged attribute values during simulation. With the “+”-button arbitrary parameters and attributes can be added. Finally a click on Next closes the Component Wizard and creates the new component.

After the completion of the Component Wizard the new component is listed in

the Modepanel in the corresponding category and can be used in a scenario.

g. Error messages and troubleshooting

<b>Error messages</b>	<b>Description</b>
Component couldn't be created - invalid connection	You tried to place the element above another element or connection. Please place the element into a free area in the scenario.
No elements to connect	No elements for a connection have been found.
Line length must be longer than 0 km	You defined a Line length of 0 km.
Invalid connection	You tried to create an invalid connection.
Start time before end time	The simulation start time has to be before the simulation end time.
Simulation range less than simulation time and greater than 0	You chose a simulation range, which is greater than the simulation time
Length of selected Line is too long	Due to the Line length and the electric power to transfer the simulation canceled. Please adjust the Line length or the Line type.

h. Shortcuts

Control key Windows/Linux: Ctrl

Control key Mac: cmd



<b>Shortcut</b>	<b>Feature</b>
Control key + ,	Settings
Control key + H	Maverig fade-out
Control key + N	New File
Control key + O	Open File
Control key + S	Save File
Control key + Shift + S	Save File as
Control key + Q	Close Maverig
Control key + Z	Undo
Control key + Y	Redo
Control key + X	Cut
Control key + Z	Copy
Control key + V	Paste
Control key + A	Select All
F1	Help
F4	Auto Layout
F5	Run Simulation
F6	Stop Simulation
F7	Pause Simulation
F8	Back to start
F9	Decrease simulation speed
F10	Increase simulation speed
F11	Forward to end
Control key + T	Set time
Control key + G	Go to
Control key + Alt + S	Selection Mode
Control key + Alt + R	Enable/Disable Raster
Control key + +	Zoom In
Control key + -	Zoom Out
Control key + .	Zoom Fit
Control key + 1	Enable/Disable Modepanel
Control key + 2	Enable/Disable Propertypanel
Control key + 3	Enable/Disable Console
Control key + 4	Enable/Disable Statusbar
Control key + 5	Enable/Disable Progressbar
Control key + 6	Enable/Disable Attributepanel

### A.3. Funktionsumfang

1. Das Produkt muss ein Ansichtsfenster bereitstellen.
2. Das Ansichtsfenster muss eine Menüleiste bereitstellen.
  - a) Die Funktionalität der Menüleiste muss jeweils durch eigens zugeordnete Tastenkombinationen ersetzbar sein.
  - b) Die Menüleiste muss ein Auswahlfenster für Dateibearbeitung bereitstellen.
    - i. Die Dateibearbeitung muss das Erstellen einer neuen Datei erlauben.
    - ii. Die Dateibearbeitung muss das Laden einer Datei erlauben.
    - iii. Die Dateibearbeitung muss das Speichern an einem angegebenen Ort erlauben.
    - iv. Die Dateibearbeitung muss das Schnellspeichern einer Datei erlauben.
    - v. Die Dateibearbeitung muss das Verlassen des Programms erlauben.
    - vi. Die Dateibearbeitung muss das Vornehmen von allgemeinen Einstellungen erlauben.
      - A. Die allgemeinen Einstellungen müssen die Auswahl der englischen Sprache ermöglichen.
      - B. Die allgemeinen Einstellungen müssen die Auswahl der französischen Sprache ermöglichen.
      - C. Die allgemeinen Einstellungen müssen die Auswahl der deutschen Sprache ermöglichen.
      - D. Die allgemeinen Einstellungen müssen die Auswahl der spanischen Sprache ermöglichen.
    - vii. Die Dateibearbeitung muss das Vornehmen von Simulationseinstellungen erlauben.
      - A. Die Simulationseinstellungen müssen die Auswahl eines Tag/Nacht-Modus für die Simulation ermöglichen.
      - B. Die Simulationseinstellungen müssen die Auswahl ermöglichen, den Einfärbungseffekt *Schatten* für Netzknoten, Leitung und Transformatoren zu wählen.

- C. Die Simulationseinstellungen müssen die Auswahl ermöglichen, den Einfärbungseffekt *Farben* für Netzknoten, Leitungen und Transformatoren zu wählen.
  - D. Die Simulationseinstellungen müssen die Auswahl ermöglichen, den Einfärbungseffekt *Schatten* für Häuser und PV-Anlagen zu wählen.
  - E. Die Simulationseinstellungen müssen die Auswahl ermöglichen, den Einfärbungseffekt *Transparenz* für Häuser und PV-Anlagen zu wählen.
  - F. Die Simulationseinstellungen müssen die Auswahl ermöglichen, die Grenzwerte über *Balken* anzuzeigen zu lassen.
- c) Die Menüleiste muss ein Auswahlfenster für Editierungen bereitstellen.
- i. Das Editierungsfenster muss das Rückgängigmachen einer Aktion ermöglichen.
  - ii. Das Editierungsfenster muss das Wiederherstellen einer Aktion ermöglichen.
  - iii. Das Editierungsfenster muss den Start einer ForceAtlas 2-Optimierung des Modellierungsgraphen ermöglichen.
  - iv. Das Editierungsfenster muss das Ausschneiden von selektierten Elementen im Darstellungsfenster ermöglichen.
  - v. Das Editierungsfenster muss das Kopieren von selektierten Elementen im Darstellungsfenster ermöglichen.
  - vi. Das Editierungsfenster muss das Löschen von selektierten Elementen im Darstellungsfenster ermöglichen.
  - vii. Das Editierungsfenster muss das Einfügen von kopierten Elementen im Darstellungsfenster ermöglichen.
  - viii. Das Editierungsfenster muss das Selektieren aller Elemente im Darstellungsfenster ermöglichen.
- d) Die Menüleiste muss ein Auswahlfenster für Simulationseinstellungen bereitstellen.
- i. Die Simulationseinstellungen müssen den Start einer Simulation ermöglichen.
  - ii. Die Simulationseinstellungen müssen den Start einer Simulationswiedergabe ermöglichen.
  - iii. Die Simulationseinstellungen müssen das Abbrechen einer Simulation ermöglichen.

- iv. Die Simulationseinstellungen müssen das Abbrechen einer Simulationswiedergabe ermöglichen.
- v. Die Simulationseinstellungen müssen das Anhalten einer Simulationswiedergabe ermöglichen.
- vi. Die Simulationseinstellungen müssen den Neustart einer Simulation ermöglichen.
- vii. Die Simulationseinstellungen müssen den Neustart einer Simulationswiedergabe ermöglichen.
- viii. Die Simulationseinstellungen müssen das Springen zu einem gewählten Zeitpunkt innerhalb einer Simulation ermöglichen.
- ix. Die Simulationseinstellungen müssen das Setzen eines Zeitfensters für eine Simulation ermöglichen.
  - A. Im Zeitfenster muss eine Startzeit angegeben werden können.
  - B. Im Zeitfenster muss eine Abschlusszeit angegeben werden können.
  - C. Im Zeitfenster muss eine Skalierung des Zeitfortschritts gewählt werden können.
  - D. Im Zeitfenster muss die Geschwindigkeit für das Voranschreiten der Zeit angegeben werden können.
- e) Die Menüleiste muss ein Auswahlfenster für die Ansicht bereitstellen.
  - i. Das Ansichtsfenster muss die Auswahl eines Shift-Modus bereitstellen.
  - ii. Das Ansichtsfenster muss die Auswahl eines Selektions-Modus bereitstellen.
  - iii. Das Ansichtsfenster muss das Heranzoomen im Darstellungsfenster ermöglichen.
  - iv. Das Ansichtsfenster muss das Herauszoomen im Darstellungsfenster ermöglichen.
  - v. Das Ansichtsfenster muss das Anpassen der Darstellungsgröße im Darstellungsfenster, gemessen an der Modellierung, ermöglichen.
  - vi. Das Ansichtsfenster muss das Ein- und Ausblenden des Modusfensters ermöglichen.
  - vii. Das Ansichtsfenster muss das Ein- und Ausblenden des Elementeigenschaftenfensters ermöglichen.

- viii. Das Ansichtsfenster muss das Ein- und Ausblenden der Konsole ermöglichen.
  - ix. Das Ansichtsfenster muss das Ein- und Ausblenden der Statusbar ermöglichen.
  - x. Das Ansichtsfenster muss das Ein- und Ausblenden der Simulationsfortschrittsanzeige ermöglichen.
  - xi. Das Ansichtsfenster muss das Ein- und Ausblenden des Attributefensters ermöglichen.
- f) Die Menüleiste muss ein Auswahlfenster für Hilfsfunktionen bereitstellen.
- i. Das Hilfsfenster muss allgemeine Hilfestellungen in Form des Benutzerhandbuchs aufrufen lassen können.
  - ii. Das Hilfsfenster muss allgemeine Informationen über die Software anzeigen lassen können.
3. Das Ansichtsfenster muss eine Toolbar bereitstellen.
- a) Die Funktionalität der Toolbar muss jeweils durch eigens zugeordnete Tastenkombinationen ersetzbar sein.
  - b) Die Toolbar muss im Programmfenster verschiebbar sein.
  - c) Die Toolbar muss das Laden einer gespeicherten Datei ermöglichen.
  - d) Die Toolbar muss das Speichern einer Datei ermöglichen.
  - e) Die Toolbar muss das Starten einer Simulation ermöglichen.
  - f) Die Toolbar muss das Starten einer Simulationswiedergabe ermöglichen.
  - g) Die Toolbar muss das Pausieren einer Simulationswiedergabe ermöglichen.
  - h) Die Toolbar muss das Abbrechen einer Simulation ermöglichen.
  - i) Die Toolbar muss das Abbrechen einer Simulationswiedergabe ermöglichen.
  - j) Die Toolbar muss das Neustarten einer Simulationswiedergabe ermöglichen.
  - k) Die Toolbar muss das Springen einer Simulationswiedergabe zum Anfang ermöglichen.
  - l) Die Toolbar muss das Springen einer Simulationswiedergabe zum Ende ermöglichen.

- m) Die Toolbar muss das Erhöhen der Wiedergabegeschwindigkeit einer Simulation ermöglichen.
  - n) Die Toolbar muss das Verringern der Wiedergabegeschwindigkeit einer Simulation ermöglichen.
  - o) Die Toolbar muss das Neustarten einer Simulation ermöglichen.
  - p) Die Toolbar muss das Neustarten einer Simulationswiedergabe ermöglichen.
  - q) Die Toolbar muss das Hereinzoomen im Darstellungsfenster ermöglichen.
  - r) Die Toolbar muss das Herauszoomen im Darstellungsfenster ermöglichen.
  - s) Die Toolbar muss das Anpassen der Darstellungsgröße im Darstellungsfenster gemessen an der Modellierung ermöglichen.
  - t) Die Toolbar muss das Löschen selektierter Elemente im Darstellungsfenster ermöglichen.
  - u) Die Toolbar muss die Simulationseinstellungen aufrufen können.
  - v) Die Toolbar muss den Start einer ForceAtlas 2-Optimierung der Modellierung ermöglichen.
4. Das Ansichtsfenster muss eine zweidimensionale Abbildungsfläche bereitstellen.
- a) Die Zeichenfläche muss eine Zoom-Funktionalität bereitstellen.
    - i. Die Zeichenfläche muss auf selektierte Elemente hin vergrößerbar sein.
    - ii. Die Zeichenfläche muss auf Mausradbedienung hin vergrößerbar sein.
    - iii. Die Zeichenfläche muss auf Mausradbedienung hin verkleinerbar sein.
    - iv. Die Zeichenfläche muss bei Verkleinerung auf die Modellierung ausgerichtet werden, wenn diese größer ist als die Fläche.
    - v. Die Zeichenfläche muss bei Verkleinerung auf die Modellierung zentrieren, wenn diese kleiner ist als die Fläche.
  - b) Die Zeichenfläche darf nicht ausblendbar sein.
  - c) Die Ansicht der Zeichenfläche muss verschiebbar sein, wenn die Modellierung größer ist als die Fläche.

5. Das Ansichtsfenster muss eine Ansicht für die Parametereinstellung markierter Elemente bereitstellen.
  - a) Die Elementparametrisierung darf nur im Kompositions-Modus aktiv sein.
  - b) Die Komponenten der Kategorien *Produzenten*, *Konsumenten*, *Speicher* und *Prosumenten* müssen im Typ durch die Zuweisung einer neuen CSV-Datei verändert werden können.
  - c) Stromleitungen müssen im Typ verändert werden können.
  - d) Stromleitungen müssen in ihrer Länge verändert werden können.
  - e) Stromleitungen müssen vom Netz genommen werden können.
  - f) Potentialknoten müssen im Spannungslevel eingestellt werden können.
  - g) Netzknoten müssen im Spannungslevel eingestellt werden können.
  - h) Transformatoren müssen im Typ verändert werden können.
  - i) Die annehmende Leitung eines Transformatoren muss eingestellt werden können.
  - j) Transformatoren müssen vom Netz genommen werden können (offline).
  - k) Bei mehreren selektierten Elementen müssen gleichzeitig Einstellungen vorgenommen werden können, wenn sie gleiche Attributtypen besitzen.
6. Das Ansichtsfenster muss eine Ansicht für die Arbeit mit Komponenten auf der Zeichenfläche bereitstellen.
  - a) Die Komponentenansicht darf nur im Kompositions-Modus aktiv sein.
  - b) Die Komponentenansicht muss eine Auswahl von Bearbeitungsmodi bereitstellen.
    - i. Ein Bearbeitungsmodus muss die Auswahl des Selektions-Modus ermöglichen, der ausschließlich das Verschieben des Darstellungsfensters erlaubt.
    - ii. Ein Bearbeitungsmodus muss die Auswahl des Auswahl-Modus ermöglichen, der die Auswahl und Bearbeitung von Komponenten erlaubt.
  - c) Die Komponentenansicht muss das Hinzufügen von Komponenten ermöglichen.
    - i. Neue Komponenten müssen mit Namen versehen werden können.
    - ii. Neue Komponenten müssen mit einer Beschreibung versehen werden können.

- iii. Neue Komponenten müssen mit einem Zeichenmodus versehen werden können.
  - A. Der Zeichenmodus muss durch eine Linie beschrieben werden können.
  - B. Der Zeichenmodus muss durch eine Linie, ein Icon und eine weitere Linie beschrieben werden können.
  - C. Der Zeichenmodus muss durch ein Icon beschrieben werden können.
  - D. Der Zeichenmodus muss durch einen Knoten beschrieben werden können.
- iv. Neue Komponenten müssen mit einer Kategorie versehen werden können.
- v. Neue Komponenten müssen mit vorgegebenen Simulatoren versehen werden können.
  - A. PyPower muss ein vorgegebener Simulator sein.
- vi. Neue Komponenten müssen mit neu definierbaren Simulatoren versehen werden können.
  - A. Neu definierbaren Simulatoren muss ein Name zugewiesen werden können.
  - B. Neu definierbaren Simulatoren muss die Startumgebung zugewiesen werden können.
  - C. Neu definierbaren Simulatoren muss eine Adresse zugewiesen werden können.
  - D. Neu definierbaren Simulatoren muss eine Menge an Parametern zugewiesen werden können.
- vii. Den Komponenten müssen eine Menge an Parametern zugewiesen werden können.
  - A. Den Komponenten muss mindestens ein Parameter zugeordnet werden.
  - B. Ein Parameter muss einen Namen besitzen.
  - C. Ein Parameter muss eine Beschreibung besitzen.
  - D. Ein Parameter muss einen Datentypen besitzen.
  - E. Ein Parameter muss eine Einschränkung akzeptierter Werte besitzen.



- F. Ein Parameter muss einen Standardwert besitzen.
- viii. Den Komponenten müssen eine Menge an Attributen zugewiesen werden können.
  - A. Den Komponenten muss mindestens ein Attribut zugeordnet werden.
  - B. Ein Attribut muss als *statisch* markiert werden können.
  - C. Ein Attribut muss einen Namen besitzen.
  - D. Ein Attribut muss eine Beschreibung besitzen.
  - E. Ein Attribut muss einen Datentypen besitzen.
  - F. Ein Attribut muss einer Einheit zugewiesen werden können.
  - G. Ein Attribut muss eine Beschreibung der Einheit besitzen.
- d) Die Komponentenansicht muss das Hinzufügen von Komponenten der Kategorie *Stromnetz* ermöglichen.
  - i. Die Ansicht zur Komponentenwahl muss die Auswahl von Potentialknoten ermöglichen.
  - ii. Die Ansicht zur Komponentenwahl muss die Auswahl von Netzknoten ermöglichen.
  - iii. Die Ansicht zur Komponentenwahl muss die Auswahl von Stromleitungen ermöglichen.
  - iv. Die Ansicht zur Komponentenwahl muss die Auswahl von Transformatoren ermöglichen.
- e) Die Komponentenansicht muss das Hinzufügen von Komponenten der Kategorie *Produzenten* ermöglichen.
  - i. Die Ansicht zur Komponentenwahl muss die Auswahl von Photovoltaikanlagen ermöglichen.
  - ii. Die Ansicht zur Komponentenwahl muss die Auswahl von Wärmekraftwerken ermöglichen.
  - iii. Die Ansicht zur Komponentenwahl muss die Auswahl von Windkraftanlagen ermöglichen.
- f) Die Komponentenansicht muss das Hinzufügen von Komponenten der Kategorie *Verbraucher* ermöglichen.
  - i. Die Ansicht zur Komponentenwahl muss die Auswahl von Verbrauchern ermöglichen.

- g) Die Komponentenansicht muss das Hinzufügen von Komponenten der Kategorie *Speicher* ermöglichen.
    - i. Die Ansicht zur Komponentenwahl muss die Auswahl von Energiespeichern ermöglichen.
  - h) Die Komponentenansicht muss das Hinzufügen von Komponenten der Kategorie *Prosumenten* ermöglichen.
    - i. Die Ansicht zur Komponentenwahl muss die Auswahl von Elektroautos ermöglichen.
7. Das Ansichtsfenster muss eine Log-Ausgabe bereitstellen.
- a) Die Log-Ausgabe muss den Start von Simulatoren ausgeben können.
  - b) Die Log-Ausgabe muss den Start der Simulation ausgeben können.
  - c) Die Log-Ausgabe muss den Abschluss der Simulation ausgeben können.
  - d) Die Log-Ausgabe muss den Fortschritt der Simulation prozentual visualisieren können.
  - e) Die Log-Ausgabe muss Fehlerquellen visualisieren können, die den Start einer Simulation verhindern.
8. Das Ansichtsfenster muss eine Fortschrittsanzeige für die Simulation bereitstellen.
- a) Die Fortschrittsanzeige darf nur im Simulations-Modus aktiv sein.
  - b) Die Fortschrittsanzeige muss den aktuellen Zeitpunkt der Simulationswiedergabe textuell darstellen.
  - c) Die Fortschrittsanzeige muss den Endzeitpunkt der Simulationswiedergabe textuell darstellen.
  - d) Die Fortschrittsanzeige muss den aktuellen Zeitpunkt der Simulationswiedergabe in Form eines Wiedergabezeigers darstellen.
  - e) Die Fortschrittsanzeige muss den aktuellen Zeitpunkt der Simulation in Form eines Ladebalkens darstellen.
  - f) Der Wiedergabezeiger darf nicht weiter als der Stand des Simulationsfortschritts angezeigt werden.
  - g) Der Wiedergabezeiger muss durch dessen Bewegung den Sprung zu einem gewählten Zeitpunkt der Simulation erlauben.
9. Das Ansichtsfenster muss eine Statusbar bereitstellen.
- a) Die Statusbar muss den aktuellen Bearbeitungsmodus der Komposition darstellen können.

- b) Die Statusbar muss darstellen können, ob der Simulations- oder Kompositions-Modus aktiv ist.
  - c) Die Statusbar muss die aktuelle Geschwindigkeit der Simulationswiedergabe wiedergeben können.
  - d) Die Statusbar muss wiedergeben können, ob die Wiedergabe der Simulation pausiert ist.
  - e) Die Statusbar muss den erfolgreichen Abschluss der Simulation anzeigen können.
  - f) Die Statusbar muss die Anschlussvalidierung gerade bewegter Elementendpunkte zu anderen Punkten anzeigen können.
  - g) Die Statusbar muss anzeigen können, dass ein Szenario erfolgreich geladen wurde.
  - h) Die Statusbar muss anzeigen können, dass ein Szenario erfolgreich gespeichert wurde.
10. Das Ansichtsfenster muss eine Attributanzeige der Elemente bereitstellen.
- a) Die Attributanzeige darf nur im Simulations-Modus aktiv sein.
  - b) Die Attributanzeige muss Attributwerte selektierter Elemente textuell als aktuellen Wert darstellen können.
  - c) Die Attributanzeige muss dynamische Attributwerte selektierter Komponenten vom Simulationsstart bis zum Zeitpunkt der Wiedergabe darstellen können.
  - d) Die Attributanzeige muss dynamische Attributwerte selektierter Komponenten als ein Liniendiagramm darstellen können.
    - i. Ein Liniendiagramm muss auf der x-Achse den Zeitwert referenzieren.
    - ii. Ein Liniendiagramm muss auf der y-Achse den Wert des entsprechenden Attributes referenzieren.
  - e) Die Attributanzeige muss Attributwerte mehrerer selektierter Elemente darstellen können, dessen Typen sie gemeinsam haben.
  - f) Die Attributanzeige muss Attributwerte mehrerer dynamischer, selektierter Komponenten in verschiedenen Farben in einem gemeinsamen Diagramm darstellen können.
  - g) Die Attributanzeige muss die Anzeige mehrerer dynamischer, selektierter Elemente im Diagramm auf zehn beschränken.

- h) Die Attributanzeige muss Liniendiagramme dynamischer Elementattribute ein- und ausblenden können.
11. Die bereitgestellten Komponenten müssen auf der Zeichenfläche editierbar sein.
- a) Ein Element-Icon der Zeichenfläche muss selektierbar sein.
  - b) Eine Element-Linie der Zeichenfläche muss selektierbar sein.
  - c) Ein Element-Linienendpunkt muss selektiert werden können.
  - d) Ein Element-Linienendpunkt darf nur selektiert werden können, wenn eine einzelne Linie ausgewählt wird.
  - e) Mehrere Elemente müssen durch Ziehen eines Rahmens mit der Maus selektiert werden können.
  - f) Elemente müssen durch das Drücken der Taste Steuerung und eines Klicks auf ein Element allesamt selektierbar sein.
  - g) Elemente des selben Typs müssen durch das zweifache Klicken auf ein Element allesamt selektierbar sein.
  - h) Alle Elemente einer Komponentengruppe müssen durch dreifaches Klicken eines Elements dieser Gruppe selektiert werden können.
  - i) Ein Element-Linienendpunkt muss verschiebbar sein.
  - j) Ein Element-Linienendpunkt muss an ein Element-Icon andockbar sein.
  - k) Ein Element-Linienendpunkt muss von einem Element-Icon abdockbar sein.
  - l) Ein selektiertes Element-Icon der Zeichenfläche muss verschiebbar sein.
  - m) Angebundene Verbindungen an verschobene Icons müssen sich parallel mit verschieben.
  - n) Selektierte Elemente müssen verschiebbar sein.
  - o) Selektierte Elemente müssen kopierbar sein.
  - p) Selektierte Elemente müssen ausschneidbar sein.
  - q) Kopierte Elemente müssen einfügbar sein.
  - r) Ausgeschnittene Elemente müssen einfügbar sein.
  - s) Selektierte Elemente müssen entfernbar sein.
12. Das Produkt muss eine Graphenoptimierung in Form von ForceAtlas 2 implementieren.

13. Die bereitgestellten Komponenten müssen in ihren Eigenschaften den Vorgaben entsprechen, die für neu zu erstellende Komponenten gelten.
14. Die bereitgestellten Komponenten müssen zusätzlich Angaben über bereits bestehende Verbindungen zu anderen Elementtypen enthalten.
15. Die bereitgestellten Komponenten müssen zusätzlich in ihren Parametern und Attributen sinnvoll grundinitialisiert sein.
16. Das Produkt muss ein Szenariomodell speichern.
  - a) Das Szenariomodell muss Informationen über den Startpunkt der Simulation speichern.
  - b) Das Szenariomodell muss Informationen über den Endpunkt der Simulation speichern.
  - c) Das Szenariomodell muss Informationen über deren Komponenteninstanzen speichern.
    - i. Die Komponenten aus der Konfiguration müssen im Modusfenster angezeigt werden können.
    - i. Die Komponenteninstanzen müssen ihre Positionierung enthalten.
    - ii. Die Komponenteninstanzen müssen ihre Parameterwerte enthalten.
    - iii. Die Komponenteninstanzen müssen ihre Verbindungsdaten enthalten.
      - i. Die Instanz einer Verbindung muss beim Verbinden zweier Elemente automatisch erzeugt werden.
      - ii. Die Komponentenkonfiguration bezüglich der Verbindungsdaten muss bei neuer Verbindung automatisch aktualisiert werden.
      - iii. Beim Versuch einer Verbindung muss dieser validiert werden.
        - A. Verbindungen dürfen nur zugelassen werden, wenn sie valide sind.
        - B. Eine Verbindung zwischen Produzentenelementen, Verbraucherelementen und Speicherelementen untereinander muss invalide sein.
        - C. Eine Verbindung zwischen Produzentenelementen, Verbraucherelementen und Speicherelementen zu Leitungen ohne Knoten muss invalide sein.

- D. Eine Verbindung zwischen Stromleitungen untereinander muss invalide sein, wenn kein Netz- oder Potentialknoten zwischen-geschaltet ist.
  - E. Eine Verbindung zwischen Stromleitungen untereinander muss invalide sein, wenn zwischen Knoten bereits eine Leitung vor-handen ist.
  - F. Eine Verbindung zwischen Transformatoren untereinander muss invalide sein, wenn kein Netz- oder Potentialknoten zwischen-geschaltet ist.
  - G. Eine Verbindung zwischen Transformatoren untereinander muss invalide sein, wenn zwischen Knoten bereits eine Verbindung vorhanden ist.
- iv. Sich überlagernde Icons müssen im Raster-Modus automatisch neu ausgerichtet werden.
17. Die Modellierung muss in jedem Schritt in einer Historie zwischengespeichert werden.
- a) Ein Modellierungsschritt muss rückgängig gemacht werden können.
  - b) Ein rückgängig gemachter Modellierungsschritt muss wiederhergestellt werden können.
18. Das Produkt muss die Mosaik-Schnittstelle anbinden können.
- a) Das Produkt muss die Simulatoren von Mosaik ermitteln können.
  - b) Das Produkt muss JSON-Dateien für die Komponenten des PyPower-Simulators erzeugen.
  - c) Das Produkt muss die Simulatoren von Mosaik starten können.
  - d) Das Produkt muss die Modellinstanzen von Mosaik entsprechend der Komponenten bilden können.
  - e) Das Produkt muss die Verbindungen entsprechend der Modellierung in Mosaik umsetzen können.
  - f) Das Produkt muss die Simulation in einem separaten Prozess starten können.
19. Im Simulations-Modus muss die Modellierung entsprechend der Komposi-tionsmodellierung erstellbar sein.
20. Die Grenzwerte der Elemente in der Simulation müssen entsprechend der ge-wählten Option dargestellt werden können.

21. Das Produkt muss einen Visualisierungssimulator zur dynamischen Datenabfrage implementieren.
  - a) Der Simulator muss eine Kommunikationsschnittstelle besitzen.
  - b) Die Kommunikationsschnittstelle muss einen Timer zur Aktualisierung der GUI implementieren.
  - c) Statische Attributwerte müssen vom Visualisierungssimulator empfangen werden können.
  - d) Dynamische Attributwerte müssen vom Visualisierungssimulator empfangen werden können.
  - e) Vom Simulator empfangene Attributwerte müssen bei den entsprechenden Komponenteninstanzen aktualisiert gesetzt werden können.
  - f) Die Heat-Parameter müssen in der Modellierungsspeicherstruktur gesetzt werden können.
  - g) Die Heat-Parameter müssen aktualisiert dargestellt werden können.
  - h) Die Simulation muss auf Basis gesammelter Daten stattfinden.