Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Michel Langhammer

Ereignisdiskreter Steuerungsentwurf für einen Nano-Grid Controller

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Michel Langhammer

Ereignisdiskreter Steuerungsentwurf für einen Nano-Grid Controller

Masterthesis eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Automatisierung
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing Florian Wenck
Zweitgutachter : Dr. Martin Jäger

Abgegeben am 02. November 2018

Michel Langhammer

Thema der Masterthesis

Ereignisdiskreter Steuerungsentwurf für einen Nano-Grid Controller

Stichworte

Ereignisdiskrete Systeme, Supervisory Control Theory, Modellbasierter Steuerungsentwurf, formale Sprachen, Automaten, Dezentrale Energiesysteme, Nano-Grid, Nanogrid

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Steuerung von einem Nanogridsystem. Es wird ein modellbasierter Steuerungsentwurf durchgeführt, welcher zur Ansteuerung von Betriebszuständen von Leistungskomponenten genutzt wird. Für die Modellbildung, Analyse und Entwurf werden Methoden der Supvisory Control Theory angewendet. Abschließend werden durch geeignete Simulationsszenarien die entwickelten Steuerungen verifiziert.

Michel Langhammer

Title of the paper

Discrete-event Control Design for a Nano-Grid Controller

Keywords

Discrete-event Systems, Supervisory Control Theory, Model-based Control Design, Formal Languages, Automata, Distributed Energy Systems, Nano-Grid, Nanogrid

Abstract

This thesis deals with the control of a nano-grid system. A model-based control design is used to control the operating states of power components. The modeling, analysis and design are based on the methods of the Supvisory Control Theory. At the end the developed supervisors are verified by suitable simulation scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AC	**A**lternating **C**urrent
CAN	**C**ommunication **A**rea **N**etwork
CP	**C**onstant **P**ower
CV	**C**onstant **V**oltage
DBS	**D**C-**B**us **S**ignaling
DC	**D**irect **C**urrent
DES	**D**iscrete-**e**vent **S**ystems (auf Deutsch: Ereignisdiskrete Systeme
DFA	**D**eterministic-**F**inite **A**utomata
GPIO	**G**eneral **P**urpose **I**nput/**O**utput
IEA	**I**nternationalen **E**nergieagentur
LGPL	**L**esser **G**eneral **P**ublic **L**icense
MPPT	**M**aximum **P**ower **P**oint **T**racking
NFA	**N**on Deterministic-**F**inite **A**utomata
PWM	**P**uls**w**eiten**m**odulation
SCT	**S**upervisory **C**ontrol **T**heory
SOC	**S**tate **o**f **C**harge
SPC	**S**trict **P**roduct **C**omposition
SYPC	**S**ynchronous **P**roduct **C**omposition

1.Einleitung

1.1 Problembeschreibung

Umweltbedenken hinsichtlich der Verbrennung fossiler Brennstoffe haben das Interesse an erneuerbaren Energien zunehmend erhöht. Eine Herausforderung bei der Umstellung von fossiler auf regenerative Energieerzeugung liegt in der fluktuierende Energieeinspeisung und der damit resultierenden Instabilität des Energienetzes. Die fluktuierende Erzeugung ist in der nicht beeinflussbaren Wetterabhängigkeit der regenerativen Quellen und dem sich stetig ändernden Verbrauch begründet. Zudem wird die Umstellung durch die Integration in eine bestehende, historisch gewachsene Netzinfrastruktur erschwert.

Laut der Internationalen Energieagentur (IEA) besitzen zudem rund 1,2 Milliarden Menschen (entspricht ca. 16% der Weltbevölkerung) keinen Zugang zu Elektrizität. Vor allem die Menschen in den Regionen in Subsahara-Afrika und Südostasien sind davon betroffen. Besonders der ländliche Raum in diesen Regionen ist meist von der öffentlichen Netzinfrastruktur ausgegrenzt. Die Elektrifizierung über das öffentliche Netz wird aufgrund von hohen Investitionskosten nicht realisiert [1].

Aus diesen Gründen werden immer mehr dezentrale Energiesysteme eingesetzt. Dabei sollen Erzeuger direkt am Verbraucher installiert und das Energiesystem möglichst autark gehalten werden. Um die Versorgung auch bei schwacher Energieerzeugung sicherzustellen, ist eine Speicherlösung im dezentralen System notwendig. Damit kann überschüssige Energie gepuffert und bei Bedarf wieder freigegeben werden.

Um einen sicheren Betrieb von dezentralen Energieanlagen zu gewährleisten ist neben der Leistungsregelung auch eine korrekte Ansteuerung der eingesetzten Komponenten notwendig. Durch den Einsatz von moderner Leistungselektronik und Kommunikationstechnik können dabei modulare und flexibel skalierbare Systemarchitekturen entworfen werden.

In der Literatur [2] [3] finden sich für den modellbasierten Entwurf eines Leistungsreglers bereits verschiedene Strategien und Methoden. Für den Entwurf einer übergeordneten Steuerung, welche alle Komponenten in einem dezentralen Energienetz ansteuert, sind hingegen wenige Ansätze gegeben. Daher soll in dieser Arbeit ein modellbasierter Steuerungsentwurf unter Verwendung ereignisdiskreter Entwurfsverfahren erstellt werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist die ereignisdiskrete Modellierung und Analyse eines Nano-Grid Systems, welches ein kleinskaliertes dezentrales Energiesystem darstellt und speziell für den Einsatz in ländlichen Gebieten konzipiert ist. Zudem soll eine Steuerungssynthese nach den Methoden der Supervisory Control Theory (SCT), welche das gegebene System durch eine definierte Spezifikation einschränkt, durchgeführt werden.

Die geforderten Steuerungsaufgaben an das Gesamtsystem sind wie folgt gegeben:

- Erzeugermanagement: korrekte Steuerung der angeschlossenen Erzeuger
- Lastmanagement: korrekte Steuerung der angeschlossenen Lasten
- Speichermanagement: korrekte Steuerung der angeschlossenen Speicher
- Sicherheitsmanagement: korrekte Steuerung einer Sicherheitskette

Das Ergebnis der Arbeit soll für jeden offen zugänglich sein, damit weiterführende Arbeiten darauf aufbauen können und auch das Wissen über dezentrale, regenerative Energiesysteme eine breite Öffentlichkeit findet. Allgemein wird damit ein Ansatz nach den Open Source Prinzipien gewählt. Auch hinsichtlich Hardware, respektive der Steuereinheiten soll auf Open Source basierte Hardware zurückgegriffen werden. Die Wahl des Entwicklungswerkzeugs wird durch die Anforderung der freien Zugänglichkeit getroffen.

1.3 Gliederung

Diese Arbeit ist in sieben Hauptkapitel gegliedert. Nach dem Einführungskapitel werden im zweiten Kapitel die Grundlagen zum Verständnis der Arbeit vermittelt. Diese setzen sich aus der Systemtheorie der ereignisdiskreten Systeme, sowie der Applikation der dezentralen Energiesysteme zusammen. Zusätzlich wird kurz das verwendete Entwicklungswerkzeug DESTool vorgestellt.

Im dritten Kapitel wird das konzipierte Nano-Grid-Modell näher erläutert. Neben den Modellkomponenten werden die Steuer- und Regelstrategien, welche zum Einsatz kommen vorgestellt.

Im vierten Kapitel wird das Entwicklungskonzept vorgestellt, in welchem verschiedene Entwicklungsumgebungen, Beschreibungsformen und Entwurfsansätze gegenübergestellt werden. Das fünfte Kapitel beinhaltet die formale Modellbildung der ungesteuerten Strecke und der Spezifikation. Der Steuerungsentwurf wird ebenfalls in diesem Kapitel beschrieben. Im sechsten Kapitel werden die entworfenen Steuerungen durch gewählte Simulationsszenarien verifiziert.

Abschließend wird eine Zusammenfassung der Arbeit durchgeführt und ein Ausblick auf weiterführende Problemstellungen gegeben.

Zum Verständnis dieser Arbeit werden Grundlagen in der Elektro- und Informationstechnik vorausgesetzt. Zusätzlich wird auf das benötigte Grundwissen in der Mengen- und Graphentheorie hingewiesen, welche im Grundlagenkapitel nicht explizit beschrieben sind.

Im Anhang ist ein elektronischer Datenträger als CD hinterlegt, auf welcher die erstellten DESTool Dateien sowie eine elektronische Kopie dieser Arbeit gespeichert sind. Diese kann bei Prof. Dr. Florian Wenck im Department Informations- und Elektrontechnik der Fakultät Technik und Informatik der HAW Hamburg eingesehen werden.

2.Grundlagen

In den nachfolgenden Kapiteln werden die Grundlagen vermittelt, welche zum Verständnis der durchgeführten Arbeit notwendig sind. Zu Beginn wird eine kurze Einführung in die allgemeine Systemtheorie gegeben. Anschließend werden die theoretischen Grundladen der ereignisdiskreten Systeme inklusive der Methoden der Supervisory Control Theory (SCT) vorgestellt. Danach werden Nano-Grids im Kontext der dezentralen Energiesysteme vorgestellt. Abschließend wird kurz auf die verwendete Entwicklungssoftware DESTool eingegangen.

2.1 Signale und Systeme

Die Systemtheorie beschäftigt sich mit der Betrachtung von realen in der Natur vorkommenden Phänomenen, auch reale Systeme genannt, und versucht diese mit Hilfe von formalen Sprachen zu formulieren. Diese Formulierung wird als Systemmodell eines realen Systems bezeichnet [4].

Diese Arbeit beschränkt sich auf reale technische Systeme. Reale technische Systeme zeichnen sich durch ihre Struktur, ihre Funktion und ihr Verhalten aus. Die Systemstruktur beschreibt die einzelnen Systemkomponenten und ihre Kopplung miteinander. Das Systemverhalten beschreibt die beobachtbare Änderung von Komponenten und dem System als Ganzes über einen bestimmten Zeitraum hinweg. Die Systemfunktion bildet die Relation zwischen dem Ziel des Anwenders und dem Systemverhalten ab [5].

Einen weiteren elementaren Bestandteil der Systemtheorie bilden Signale. Signale repräsentieren Informationen zwischen zwei Systemen. Sie werden unter Betrachtung der Zeit in kontinuierliche und diskontinuierliche Signale unterteilt. Kontinuierliche Signale werden durch stückweise stetige Funktionen $f(x)$ beschrieben. Diskontinuierliche Signale werden durch Funktionen $f[n]$ beschrieben. Dabei wird jedem ganzzahligen Wert n ein eindeutiger skalarer Wert zugeordnet. Die ganzzahligen Werte für n werden als diskrete Zeitpunkte und der zugeordnete Wert als zeitdiskretes Signal bezeichnet [4].

Anhand eines erstellten Modells können nun Eigenschaften und Verhaltensweise geprüft und analysiert werden. Der Modellbildungsprozess erfordert eine Abstraktion und Strukturierung des vorhandenen realen Systems. Der Abstraktionsgrad ist durch den Informationsgehalt der

im System enthaltenen Signale definiert. Durch Quantisierung des Wertebereichs erhält man eine höhere Abstraktionsebene. Dabei ist zu beachten, dass jeder Quantisierungsschritt auch ein Informationsverlust des betrachteten Signals bedeutet. Der Anwender ist verantwortlich dafür, welche Abstraktion gewählt wird um das Systemmodell zu erstellen und korrekt zu beschreiben. Ziel ist immer eine zweckmäßige Betrachtung sowie die Möglichkeit der Problemlösung [6].

2.2 Ereignisdiskrete Systeme

In den folgenden Kapiteln wird die Theorie der ereignisdiskreten Systeme (englisch Discrete-Event Systems, DES) erläutert. Die Methoden der DES spielen für den durchgeführten modellbasierten Steuerungsentwurf eine elementare Rolle.

2.2.1 Einführung

Ereignisdiskrete Systeme bilden eine Teilmenge von diskontinuierlichen diskreten Systemen. Sie beschreiben ein dynamisches System mit diskretem Zustandsraum. Anders als bei zeitdiskreten Signalen, welche über die Zeit gesteuert werden, werden die Signale durch Ereignisse gesteuert. Ein Ereignis ist eine plötzlich auftretende Änderung eines Eingangs-, Ausgangs- oder Zustandssignals. Idealisierend betrachtet nimmt diese Änderung keine Zeit in Anspruch. Beispiel für ein Ereignis ist das Starten eines Prozesses oder das plötzliche Auftreten eines Fehlers. Ein Signal hat nach einem abrupten Signalwechsel einen konstanten Wert, welcher zu einer endlichen Menge an diskreten Signalwerten bzw. Ereignissen gehört [6]. In der folgenden Abbildung werden zur Veranschaulichung die verschiedenen Signalarten gegenübergestellt.

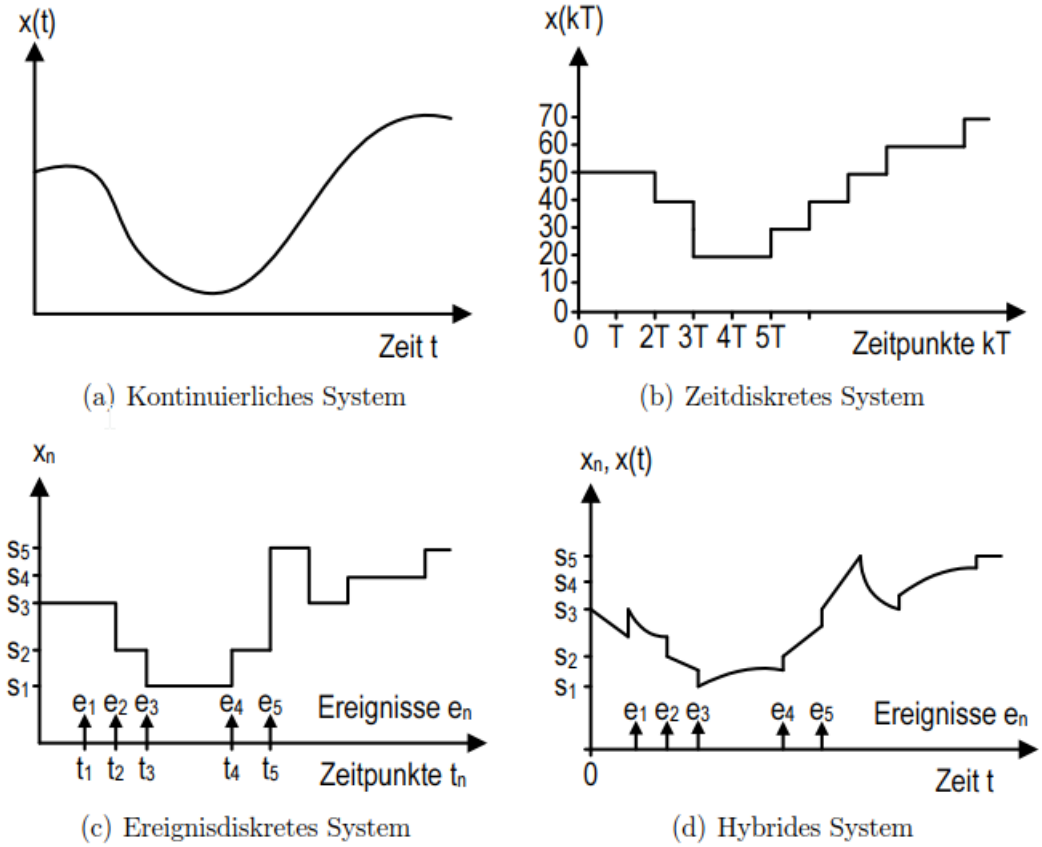

Abbildung 1: Zusammenhang zwischen verschiedenen Signalarten [7]

Der Unterschied zwischen kontinuierlichen und ereignisdiskreten Systemen liegt im Wertebereich der Signale. Kontinuierliche Signale haben einen reellen Wertebereich mit unendlich vielen möglichen Signalwerten (siehe stetigen Kurvenverlauf (a), Abbildung 1) . Diskrete Signale besitzen dagegen nur endlich viele einzelne Werte (siehe (b) und (c), Abbildung 1). Eine Kombination aus beiden Wertebereichen findet sich in hybriden Systemen wieder (siehe (d), Abbildung 1) [7].

Um das Verhalten ereignisdiskreter Systeme beschreiben und analysieren zu können, müssen Modelle entwickelt werden, die die asynchrone Arbeitsweise von Prozessen und den eventuell enthaltenen Teilprozessen sowie den Informationsaustausch durch Ereignisse, abbilden können. Diese Modelle sind in der Theorie der ereignisdiskreten Systeme vorhanden. Die Beschreibung erfolgt über eine Funktion von Ereignisketten. Ein System erfüllt eine korrekte Funktion, wenn die Ereignisse in der gewünschten Reihenfolge auftreten. Anders als bei vielen eingesetzten ingenieurwissenschaftlichen Methoden wird ein Systemverhalten nicht durch Differentialgleichungen und algebraischen Gleichungen formuliert, sondern anhand Graphentheoretischer Verfahren [6].

Der Grund für eine ereignisdiskrete Betrachtung liegt zum einen im Charakter der ablaufenden Prozesse zum anderen im Modellbildungsziel. In Fertigungsprozessen liegt das

physikalische System einer Montagelinie bereits in einem diskreten Wertebereich vor. Bei kontinuierlichen Prozessen kann durch Abstraktion ein diskreter Signalraum gebildet werden. Bei der Steuerung von einzelnen Teilprozessen eines Gesamtsystems ist meist nicht der kontinuierliche Verlauf jedes einzelnen Teilprozesses relevant sondern das Zusammenwirken der einzelnen Prozesse miteinander. Durch die Abstraktion wird die Behandlung eines Systems auf das für die Lösung einer Aufgabe Wesentliche konzentriert [6].

Um aus kontinuierlichen Signalen aus einem technischen Prozess diskrete Ereignissignale zu generieren ist die Integration eines Ereignisgenerators notwendig, wie in folgender Abbildung zu erkennen ist.

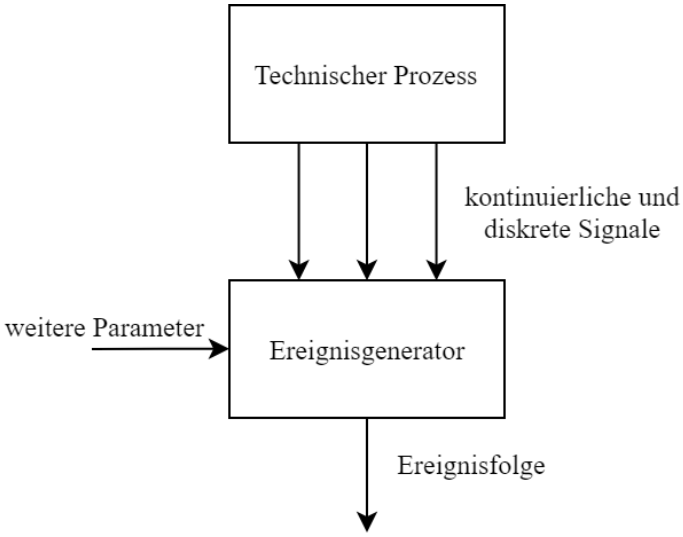

Abbildung 2: Integration eines Ereignisgenerators (angepasst um den Pfeil *weitere Parameter* von [6])

Wertekontinuierlichen Signale wie beispielsweise Strom oder Spannung werden als Eingangsvariablen gelesen und durch Operationen zusammengefasst. Ein Beispiel für eine einfache Operation ist eine Schwellwertbildung. Hat ein kontinuierliches Signal eine Wertgrenze über- oder unterschritten, wird ein binäres Signal ausgegeben. Es können auch komplexere Operationen angewendet werden, welche um externe Funktionsparameter erweitert sind. Der Ereignisgenerator ist eine elementare Schnittstelle zwischen dem realen technischen Prozess und der ereignisdiskret entworfenen Steuerung.

2.2.2 Sprachentheoretische Grundlagen

Die Beschreibung von ereignisdiskreten Systemen erfolgt über reguläre Sprachen, welche Teil der formalen Sprachen sind. Reguläre Sprachen wurden 1958 von Noam Chomsky hierarchisch nach ihrer Ausdrucksmächtigkeit, welche über die enthaltene Grammatik

bestimmt wird, klassifiziert [7]. Zum Verständnis der erstellten ereignisdiskreten Modelle werden die benötigten sprachentheoretischen Grundlagen erläutert. Auf die Vorstellung der mengentheoretischen Grundlagen wird in dieser Arbeit verzichtet. Diese werden als bekannt angenommen. In folgender Auflistung werden die grundlegenden Begriffe, Definitionen und Operationen aufgeführt. Diese sind aus [8] und [9] entnommen.

- Ein Symbol σ ist ein Zeichen und kann ein Ereignis darstellen
- Ein Alphabet Σ beschreibt eine endliche nichtleere Menge von paarweise verschiedenen Symbolen
- Ein String s ist eine endliche Sequenz von Symbolen $\sigma_1\sigma_2\sigma_3\ldots\sigma_k$
- ε ist eine Sequenz ohne Symbole mit $\varepsilon \notin \Sigma^+$, $\varepsilon \notin \emptyset$
- Σ^+ beschreibt die Menge aller möglichen Strings bzw. Ereignisfolgen aus Σ, ausgenommen ε
- Die Kleene-Hülle Σ^* von Σ ist die Menge aller möglicher Ereignisfolgen inklusive ε und wird definiert als $\Sigma^* = \{\varepsilon\} \cup \Sigma^+$
- Die Konkatenation cat ist definiert als $cat(\varepsilon, s) = cat(s, \varepsilon) = s$ mit $s \in \Sigma^*$ und $cat(s, t) = st$ mit $s, t \in \Sigma^*$
- Ein Präfix von s ist $\bar{s} = t$, wenn $s = cat(t, \sigma) = t\sigma$ mit $t \in \Sigma^*$und $\sigma \in \Sigma$
- Ein Suffix eines Strings s ist t ab σ, wenn $s = u\sigma t$ mit $u, t \in \Sigma^*, \sigma \in \Sigma$
- t ist ein Substring von s für $s = utv$ mit $u, t, v \in \Sigma^*$
- $|s|$ beschreibt die Länge eines Strings und ist definiert als $|\varepsilon| = 0$, $|s| = k$ für $s = \sigma_1 \ldots \sigma_k \in \Sigma^+$

Über einem Alphabet Σ ist die erzeugte formale Sprache L als eine beliebige Teilmenge $L \in \Sigma^*$ definiert. Als reguläre Sprache werden formale Sprachen bezeichnet, die von endlichen Automaten erkannt werden. Sprachen können mit den grundlegenden Operationen, wie Vereinigung, Schnitt, Differenz, Komplement und symmetrische Differenz verknüpft werden. Zusätzlich sind folgende Operationen auf reguläre Sprachen anwendbar.

- Konkatenation: Die Konkatenation zweier Sprachen $L_1, L_2 \subseteq \Sigma^*$enthält die zusammengesetzten Strings aus L_1 und L_2und ist definiert als:

$$L_1L_2 = \{s \in \Sigma^* | (s = s_1s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2)\} \tag{2.1}$$

- Präfix-Hülle: In der Präfix-Hülle $\bar{L}$ einer Sprache $L \subseteq \Sigma^*$ sind alle Präfixe aller Strings $s \in L$ enthalten. Sie ist definiert als:

$$\bar{L} = \{s \in \Sigma^* | \exists t \in \Sigma^* \ (st \in L)\} \tag{2.2}$$

Eine Sprache ist präfix-geschlossen, wenn $L = \bar{L}$ gilt.

- Kleene-Hülle: Die Kleene L^* enthält alle Strings, die durch Konkatenation einer endlichen Anzahl von Strings der Sprache L entstehen. Zusätzlich enthält sie den leeren String ε. Für $L \subseteq \Sigma^*$ ist L^* definiert als:

$$L^* = \{\varepsilon \cup L \cup LL \cup LLL \cup \ldots\} \tag{2.3}$$

Die regulären Sprachen sind unter den Operationen Vereinigung, Schnitt, Komplement, Konkatenation, Bildung der Präfix- und der Kleene-Hülle abgeschlossen. Das Ergebnis der Operationen bildet damit wieder eine reguläre Sprache. Diese Abschlusseigenschaft ist für die Komposition und Analyse von logischen DES, sowie deren Steuerungsentwurf von Bedeutung [7].

2.2.3 Automaten als Beschreibungsform

Um das dynamische Systemverhalten ereignisdiskreter Systeme zu beschreiben, können nach [6] logische, zeitbewertete, stochastische und zeitbewertete stochastische Modelle verwendet werden. Diese Arbeit beschränkt sich auf nicht zeitbewertete ereignisdiskrete Systeme, welche anhand logischer Modelle beschrieben werden. Es wird davon ausgegangen, dass ein Systemzustand beliebig lange aktiv sein kann. Logische Modelle bilden die grundlegende Modellform ereignisdiskreter Systeme. Das logische Verhalten wird als eine Bewegung endlicher Mengen diskreter Zustände beschrieben. Logische Modelle lassen erkennen welche Ereignisse in welcher Reihenfolge eintreten.

Die logische Beschreibung des erstellten Modells wird in dieser Arbeit durch endliche deterministische Automaten (englisch: Deterministic-Finite Automata, DFA) durchgeführt. Im Gegensatz zu endlichen nicht deterministischen Automaten (englisch: Non-Deterministic-Finite Automata, NFA) besitzen DFA eine tatsächliche Funktion, die jedem Ereignis einen eindeutigen Funktionswert zuordnet. Technische Systeme zeichnen sich durch deterministisches Verhalten aus. Automaten beschreiben die Systemstruktur durch Zustände und Transitionen. Das Systemverhalten wird durch die generierte Sprache beschrieben.

In dieser Arbeit wird als Automatenform das Generatorkonzept von Wohnham und Ramadge angewendet [9] [10]. Ein Generator G wird durch das 5-Tupel

$$G = (X, \Sigma, \delta, x_0, X_m) \tag{2.4}$$

beschrieben. Die Menge X stellt die endliche Zustandsmenge des Generators dar. Das Ereignisalphabet, welches die Menge aller möglichen Ereignisse beinhaltet, wird in Σ angegeben. Anhand der Zustandsübergangsfunktion δ wird das dynamische Verhalten des Generators beschrieben und stellt üblicherweise eine partielle Übergangsfunktion dar. Die Zustandsübergangsfunktion δ ist definiert als

$$\delta = X \times \Sigma \to X. \tag{2.5}$$

Jedem Zustand $x \in X$ wird durch δ ein Folgezustand $x' \in X$ zugeordnet, welcher durch das Auftreten eines Ereignisses $\sigma \in \Sigma$ erreicht wird und mit

$$x' = \delta(x, \sigma), x' \in X \tag{2.6}$$

definiert. Die Existenz einer Übergangsfunktion für ein Ereignis σ im aktuellen Zustand x wird durch ein Ausrufezeichen mit $\delta(x, \sigma)!$ signalisiert. Eine Nichtexistenz wird mit der Schreibweise $\neg\delta(x, \sigma)!$ beschrieben. Wenn der existierende Folgezustand den aktuellen Zustand mit $x = \delta(x, \sigma)$ darstellt, wird der Übergang als Schlinge bezeichnet. Eine Erweiterung der Zustandsübergangsfunktion für Strings erfolgt nach [11] und ist für $s \in \Sigma^*$ und $\sigma \in \Sigma$ definiert als

$$\delta(x, \varepsilon) = x, \tag{2.7}$$

$$\delta(x, s\sigma) = \delta(\delta(x, s), \sigma). \tag{2.8}$$

Hierbei wird vorausgesetzt, dass $\delta(x, s)!$ und $\delta(x, \sigma)!$ erfüllt ist. Jeder Generator benötigt einen definierten Initialzustand, in welchem sich der Generator ohne Auftreten eines Ereignisses zu Beginn befindet. Dieser wird mit dem Anfangszustand $x_0 \in X$ angegeben. Die Menge der markierten Zustände X_m mit $X_m \subseteq X$ gibt die Zielzustände bzw. akzeptierten Zustände des Systems an. Diese können einen abgeschlossenen Prozess oder das Erreichen eines bestimmten Betriebszustands darstellen.

Zur verbesserten Darstellung der Zustands-Transitionsstruktur eines Generators werden gerichtete Graphen verwendet. Zustände werden über Knoten, Transitionen über gerichtete Kanten abgebildet. Folgende Abbildung zeigt einen gerichteten Beispielgraphen $X = \{1,2,3\}$, $\Sigma = \{a, b, c, d, e\}$, $x_0 = 1$, $X_m = \{3\}$.

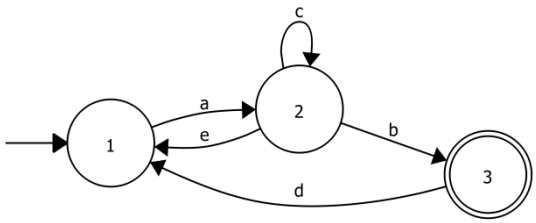

Abbildung 3: Generatordarstellung als gerichteter Graph

Der Anfangszustand wird mit einem eingehenden Pfeil und der markierte Zustand mit einem Doppelkreis gekennzeichnet. Neben der grafischen Darstellung des Generators kann dieser auch in Matrixform beschrieben werden. Dazu wird eine Adjazenzmatrix A mit $N \times N$ Elementen gebildet, wobei N die Anzahl der Zustände beschreibt. Ist die Übergangsfunktion δ von j nach i definiert, so ist das Matrixelement a_{ij} gleich dem Ereignis σ, sonst 0. Die Adjazenzmatrix A für den in Abbildung 3 dargestellten Generator ist demnach mit

$$A = \begin{pmatrix} 0 & e & d \\ a & c & 0 \\ 0 & b & 0 \end{pmatrix} \tag{2.9}$$

definiert. Vorhandene Schlingen werden in den Diagonalelementen der Matrix abgebildet [6].

Das dynamische ungesteuerte Systemverhalten eines logischen ereignisdiskreten Systems wird anhand der regulären Sprache $L(G) \in \Sigma^*$ beschrieben. $L(G)$ bildet die Gesamtheit aller von einem Generator G generierbaren Strings ab. Diese stellen die Konkatenation der Ereignisse aller möglichen gerichteten Pfade, die vom Anfangszustand x_0 aus durchlaufen werden können, dar und ist nach [7] definiert als

$$L(G) = \{s \in \Sigma^* | \delta(x_0, s)!\}. \tag{2.10}$$

Aufgrund der historischen Entwicklung von $L(G)$ existiert ein String s nur, wenn auch alle seine Präfixe existieren. Damit ist $L(G)$ immer präfix geschlossen und es gilt $L(G) = \overline{L(G)}$.

Mit $L_m(G) \subseteq L(G)$ wird die reguläre Sprache bezeichnet, die alle Strings $s \in L(G)$ enthält, die zu einem markierten Zustand $x \in X_m$ führen. Mit $L_m(G)$ wird das markierte Verhalten von G beschrieben und ist nach [7] definiert als

$$L_m(G) = \{s \in \Sigma^* | \delta(x_0, \mathrm{s}) \in X_m\}. \tag{2.11}$$

Für $X = X_m$ gilt $L(G) = L_m(G) = \overline{L(G)} = \overline{L_m(G)}$. Diese Eigenschaft wird für den späteren Steuerungsentwurf genutzt.

Neben der formalen Beschreibung durch Automaten, können auch Petrinetze verwendet werden. Diese werden in dieser Arbeit nicht angewendet und daher nicht weiter betrachtet. Es wird auf die Literatur [12] verwiesen.

2.2.4 Komposition

Ein reales System als Ganzes in monolithischer Form zu beschreiben ist bei der heutigen technischen Komplexität sehr schwierig und meist nicht zielführend. Daher wird ein System in einzelne Funktionsblöcke partitioniert. Um diese Partitionierung in ein ereignisdiskretes logisches Modell zu fassen, werden die Systemkomponenten in einzelne Generatoren gefasst und anschließend durch eine Komposition wieder zusammengeführt. Dieser Vorgang wird als kompositionale Modellbildung bezeichnet und lässt eine strukturierte Beschreibung in ein unstrukturiertes Gesamtmodell überlaufen. Kompositionsoperatoren sind in dieser Arbeit immer als binäre Operatoren definiert. Werden mehr als zwei Generatoren miteinander gekoppelt, ist die Operation sukzessive auszuführen. Nachfolgend werden zwei Kompositionsoperatoren vorgestellt, die für diese Arbeit relevant sind. Die verwendeten Definitionen sind aus [6], [7] und [13] entnommen.

Es werden zwei Generatoren G_1 und G_2 mit ihren Ereignisalphabeten Σ_1 und Σ_2 betrachtet. Die Menge aller möglichen Ereignisse $\Sigma = \Sigma_1 \cup \Sigma_2$ wird in private Ereignisse Σ_{pe} (englisch: private event, pe) und gemeinsame Ereignisse Σ_{ce} (englisch: common event, ce) mit

$$\Sigma_{ce} = \Sigma_1 \cap \Sigma_2 \text{ und} \tag{2.12}$$

$$\Sigma_{pe} = (\Sigma_2 - \Sigma_1) \cup (\Sigma_1 - \Sigma_2) \tag{2.13}$$

unterteilt.

Die **Produktkomposition**, welche auch strenge Synchronisation (englisch: Strict Product Composition, SPC) genannt wird, synchronisiert zwei Generatoren G_1 und G_2 auf ihre gemeinsamen Ereignisse. Private Ereignisse werden bei diesem Operator nicht zugelassen. Die Produktkomposition für G_1 und G_2 mit $x_1 \in X_1$, $x_2 \in X_2$ und $\sigma \in \Sigma_{ce}$ ist definiert als

$$G = G_1 ||_{SPC} G_2 = (X_1 \times X_2, \Sigma_{ce}, \delta, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \text{ mit} \tag{2.14}$$

$$\delta((x_1, x_2), \sigma) = \begin{cases} \delta_1(x_1, \sigma) \times \delta_2(x_2, \sigma) & falls\ \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)!. \\ nicht\ definiert & sonst. \end{cases} \tag{2.15}$$

Im synchron gekoppelten Generator G werden somit die privaten Ereignisse Σ_{pe} ausgeblendet und ein Zustandsübergang kann nur dann erfolgen, wenn beide Komponenten G_1 und G_2 das Ereignis in ihrem jeweiligen Zustand generieren können. Es wird auch der Begriff des totalen Gleichschritts verwendet. Die erzeugten regulären Sprachen $L(G)$ und $L_m(G)$ enthalten folglich nur gemeinsame Strings aus G_1 und G_2 und sind beschrieben als

$$L(G) = L(G_1||_{SPC}G_2) = L(G_1) \cap L(G_2) \tag{2.16}$$

$$L_m(G) = L_m(G_1||_{SPC}G_2) = L_m(G_1) \cap L_m(G_2). \tag{2.17}$$

In der praktischen Anwendung wird dieser Operator für das Zusammenschalten von Steuerung und Strecke genutzt.

Die **Parallele Komposition**, welche auch als synchrones Produkt (englisch: Synchronous Product, SYPC) bezeichnet wird, synchronisiert zwei Generatoren G_1 und G_2 auf ihre gemeinsamen Ereignisse und lässt zusätzlich private Ereignisse zu. Damit ist ein asynchrones Schalten der Automaten möglich. Das synchrone Produkt für G_1 und G_2 mit $x_1 \in X_1$, $x_2 \in X_2$ und $\sigma \in \Sigma$ ist definiert als

$$G = G_1||_{SYPC}G_2 = (X_1 \times X_2, \Sigma, \delta, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \text{ mit} \tag{2.18}$$

$$\delta((x_1, x_2), \sigma) = \begin{cases} \delta_1(x_1, \sigma) \times \delta_2(x_2, \sigma) & falls\ \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)!. \\ \delta_1(x_1, \sigma) \times x_2 & falls\ \delta_1(x_1, \sigma)! \wedge \neg\delta_2(x_2, \sigma)! \wedge \sigma \notin \Sigma_{ce}. \\ x_1 \times \delta_2(x_2, \sigma) & falls\ \neg\delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)! \wedge \sigma \notin \Sigma_{ce}. \\ nicht\ definiert & sonst. \end{cases} \tag{2.19}$$

Im parallel gekoppelten Generator G findet ein Zustandsübergang nur dann statt, wenn beide Komponenten das Ereignis im jeweiligen Zustand ausführen können oder wenn nur eine Komponente ein Ereignis ausführen kann, welches nicht zum gemeinsamen Ereignisalphabet Σ_{ce} gehört. Dies führt zu einem partiellen Gleichschritt der beiden Generatoren. Besitzen die Komponenten keine gemeinsamen Ereignisse, bewegen sie sich komplett unabhängig voneinander. Dieser Fall wird als Shuffle-Produkt bezeichnet. Besitzen die Komponenten ausschließlich gemeinsame Ereignisse und keine privaten Ereignisse, so geht die Parallele Komposition in die Produktkomposition über. Dieser Fall wird als Meet bezeichnet. Eine praktische Anwendung des SYPC Operators ist die Ressourcenzuteilung in Anlagen oder das Zusammenschalten von einzelnen Anlagenmodulen zu einem Gesamtsystem.

Die erzeugte reguläre Sprache des parallel gekoppelten Generators ist definiert als

$$L(G) = L(G_1||_{SYPC}G_2) = {P_{\Sigma_1}}^{-1}(L(G_1)) \cap {P_{\Sigma_2}}^{-1}(L(G_2)). \quad (2.20)$$

Der vorhandene Operator P^{-1} beschreibt die inverse natürliche Projektion. Die natürliche Projektion $P: \Sigma^* \to \Sigma_1^*$ ist eine Abbildung und entfernt aus einem String $s \in \Sigma^*$ alle Ereignisse $\sigma \in \Sigma - \Sigma_1$. Es gilt für die natürliche Projektion

$$P(\varepsilon) = \varepsilon, \quad (2.21)$$

$$P(\sigma) = \begin{cases} \sigma & falls\ \sigma \in \Sigma_1, \\ \varepsilon & sonst, \end{cases} \quad (2.22)$$

$$P(s\sigma) = P(s)P(\sigma) \text{ für } s \in \Sigma^*, \sigma \in \Sigma. \quad (2.23)$$

Mit der Definition der natürlichen Projektion ist die inverse Projektion $P^{-1}: \Sigma_1^* \to 2^{\Sigma^*}$ definiert als

$$P^{-1}(t) = \{s \in \Sigma^* | P(s) = t\}. \quad (2.24)$$

Die Menge 2^{Σ^*} ist eine Potenzmenge von Σ^* und bildet die Menge aller Teilmengen. Auf einen Generator angewendet, wird für jedes Ereignis $\sigma \in \Sigma - \Sigma_1$ eine Schlinge an jeden Zustand gesetzt.

2.2.5 Analyse ereignisdiskreter Systeme

Anhand der formalen Beschreibung durch Automaten und regulären Sprachen lassen sich ereignisdiskrete Systeme unter Verwendung der in diesem Abschnitt erläuterten Methoden aus [6] und [7] analysieren.

Die **Erreichbarkeits- und Ko-Erreichbarkeitsanalyse** beantwortet die Frage welche Systemzustände vom Anfangszustand ausgehend erreichbar sind und, ob das System aus einem beliebigen Zustand heraus einen Zielzustand erreichen kann. Ein Zustand $x \in X$ ist erreichbar, wenn ein String $s \in \Sigma^*$ existiert, sodass $\delta(x_0, s)!$ und $\delta(x_0, s) = x$. Für die Erreichbarkeitsanalyse wird die Ac-Operation (vom englischen Begriff accessible abgeleitet) angewendet, welche einen Generator G auf seine erreichbare Zustandsmenge $X_{ac} \subseteq X$ reduziert. Die Ac-Operation ist definiert als

$$Ac(G) = \left(X_{ac}, \Sigma, \delta_{ac}, x_0, X_{ac,m}\right), \text{ mit} \quad (2.25)$$

$$X_{ac} = \{x \in X | \exists s \in \Sigma^* \ (\delta(x_0, s) = x)\}, \quad (2.26)$$

$$X_{ac,m} = X_m \cap X_{ac}, \tag{2.27}$$

$$\delta_{ac} = \delta|_{X_{ac}\times\Sigma\to X_{ac}}. \tag{2.28}$$

Der Anfangszustand x_0 und das Ereignisalphabet Σ bleiben unverändert. Die Notation $\delta|_{X_{ac}\times\Sigma\to X_{ac}}$ schränkt δ auf die erreichbaren Zustände ein. Ein Generator ist *erreichbar*, genau dann wenn $G = Ac(G)$. Die Ac-Operation hat keinen Einfluss auf die Sprachen $L(G)$ und $L_m(G)$.

Anhand der Ko-Erreichbarkeitsanalyse wird ein Generator G auf Zustände untersucht, von denen aus mindestens ein String $s \in \Sigma^*$ existiert, der zu einem markierten Zustand führt, sodass $\delta(x, s) \in X_m$. Bei Existenz eines solchen Strings ist der Zustand $x \in X$ ko-erreichbar. Die $CoAc$-Operation (vom englischen Begriff co-accessible abgeleitet) reduziert den Generator auf seine ko-erreichbaren Zustände X_{coac}. Die $CoAc$-Operation ist defniert als

$$CoAc(G) = \left(X_{coac}, \Sigma, \delta_{coac}, x_{coac,0}, X_m\right), \text{ mit} \tag{2.29}$$

$$X_{coac} = \{x \in X | \exists s \in \Sigma^*(\delta(x, s) = X_m)\}, \tag{2.30}$$

$$x_{coac,0} = \begin{cases} x_0 & falls\ x_0 \in X_{coac}, \\ nicht\ definiert & sonst, \end{cases} \tag{2.31}$$

$$\delta_{coac} = \delta|_{X_{coac}\times\Sigma\to X_{coac}}. \tag{2.32}$$

Die Menge der markierten Zustände X_m und das Ereignisalphabet Σ bleiben unverändert. Die Notation $\delta|_{X_{coac}\times\Sigma\to X_{coac}}$ schränkt δ auf die ko-erreichbaren Zustände ein. Ein Generator ist genau dann *ko-erreichbar*, wenn $G = CoAc(G)$. Durch die mögliche Entfernung von erreichbaren Zustände, hat die $CoAc$-Operation Auswirkung auf die Sprache $L(G)$. Das markierte Verhalten $L_m(G)$ bleibt hingegen unverändert. Für einen ko-erreichbaren Generator G gilt $L(G) = \overline{L_m(G)}$. Es werden demnach nur Strings generiert, die zu einem markierten Zustand führen.

Die sequentielle Ausführung der beiden Operationen Ac und $CoAc$ bildet die $Trim$-Operation. Diese reduziert einen Generator auf seine erreichbaren und ko-erreichbaren Zustände und ist kommutativ definiert als

$$Trim(G) = CoAc[Ac(G)] = Ac[CoAc(G)]. \tag{2.33}$$

Ein Generator ist genau dann *trim*, wenn $G = Trim(G)$.

Zusätzlich zu den vorgestellten Ereichbarkeits- und Ko-Erreichbarkeitsanalysen, kann ein Generator hinsichtlich **Deadlock, Livelock und Blockierung** überprüft werden. Anhand dieser Eigenschaften wird untersucht, ob ein Prozess zweckmäßig beendet werden kann. Ein Deadlock ist dann vorhanden, wenn ein Generator G einen erreichbaren Zustand $x \notin X_m$ mit $\neg\delta(x,\sigma)!$ und $\forall\sigma \in \Sigma$ enthält. Wird ein solcher Zustand erreicht, können keine weiteren Ereignisse generiert werden. Damit kann insbesondere kein markierter Zustand erreicht werden. Existiert ein Deadlock, so gilt $\overline{L_m(G)} \subset L(G)$.

Ein Livelock liegt dann vor, wenn ein Generator G eine Zustandsmenge $\tilde{X} \subseteq X$ mit $x \notin X_m$ und $\forall x \in \tilde{X}$ erreicht, von der aus kein markierter Zustand mehr erreicht werden kann. Existiert ein Livelock, gilt ebenfalls $\overline{L_m(G)} \subset L(G)$.

Deadlock und Livelock haben gemeinsam, dass kein markierter Zustand erreicht werden kann. Ein Generator ist blockierend, sobald er mindestens ein Deadlock oder ein Livelock enthält. Da ein markierter Zustand die Abarbeitung eines gewünschten Prozesses darstellt, können blockierende Generatoren niemals diesen Prozessschritt erreichen. Formal ist ein Generator blockierend, wenn $\overline{L_m(G)} \subset L(G)$ und nicht blockierend, wenn

$$\overline{L_m(G)} = L(G). \tag{2.34}$$

In technischen Prozessen ist das Kriterium des Nichtblockierens von großer Bedeutung und wird auch in dieser Arbeit als Analyseschritt im durchgeführten Steuerungsentwurf verwendet.

2.2.6 Supervisory Control Theory

Die Supervisory Control Theory (SCT) beinhaltet eine mathematische Methodensammlung zur formalen Modellierung, Analyse sowie zur systematischen Synthese von Steuerungen für ereignisdiskrete Systeme. Sie kann sowohl für logische als auch zeitbewertete Modelle angewendet werden. In dieser Arbeit werden nur die Methoden für logische DES betrachtet. Die grundlegenden Ideen der SCT wurden erstmals in der Dissertation von P.J. Ramadge [14] aufgezeigt. Aus diesen sind zahlreiche Publikationen zu diesem Thema hervorgegangen [15], [10] und [16]. In industriellen sowie energietechnischen Applikationen wird das Konzept der SCT noch wenig angewendet. Dies liegt vor allem in der geringen Verbreitung der formalen Beschreibungsform durch Generatoren und regulären Sprachen. Aus diesen Gründen wurden die Methoden der SCT überwiegend an kleinen theoretischen Beispielen untersucht. Ein Ziel dieser Arbeit ist die Anwendbarkeit für dezentrale Energiesysteme darzustellen.

Grundlegend wird für den Einsatz der SCT das betrachtete System in eine zu steuernde Strecke G und eine Spezifikation K aufgeteilt. Die Strecke beschreibt das ungesteuerte Verhalten eines Systems. Die Spezifikation stellt das gewünschte Verhalten des Systems dar. Beide Komponenten werden als Generatoren modelliert. Aus beiden Modellen wird anschließend der Supervisor S synthetisiert. Dieser beeinflusst die Steuerstrecke G durch das Erlauben und Verbieten bestimmter Signale und führt so zum gewünschten Systemverhalten. In realen technischen Systemen werden Signale in Aktoren und Sensoren unterteilt, welche in der SCT als Ereignisse interpretiert werden. Sensoren sind eingehende, nicht beinflussbare Signale. Aus diesem Grund wird das Ereignisalphabet Σ der Systemkomponenten G in die disjunkten Ereignisalphabete der steuerbaren Ereignisse Σ_c (Index c wird aus dem englischen Begriff controllable abgeleitet) und nicht steuerbaren Ereignisse Σ_{uc} (Index uc wird aus dem englischen Begriff uncontrollable abgeleitet) unterteilt und sind nach [9]definiert als

$$\Sigma = \Sigma_c \cup \Sigma_{uc} \text{ mit } \Sigma_c \cap \Sigma_{uc} = \emptyset. \tag{2.35}$$

Die gewünschte Systemfunktion kann damit nur durch das Erlauben und Verbieten von steuerbaren Ereignissen erfolgen.

Durch Zusammenschalten der synthetisierten Steuerung S und der ungesteuerten Strecke G wird das gewünschte gesteuerte Systemverhalten anhand des geschlossenen Steuerkreis S/G repräsentiert. Folgende Abbildung zeigt das Blockschaltbild von S/G.

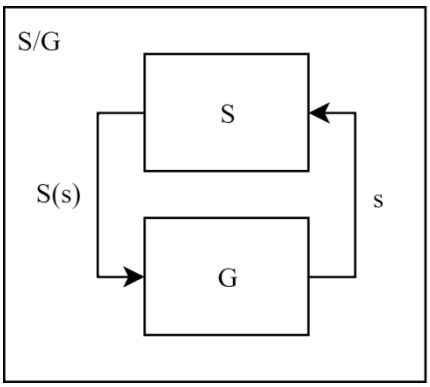

Abbildung 4: Blockschaltbild des monolithisch geschlossenen Steuerkreis S/G

Die von der Strecke G generierte Ereignisfolge s wird an den Supervisor S übergeben. Dieser überwacht die Ereignisfolge und deaktiviert die unerwünschten steuerbaren Ereignisse. In der Ereignismenge $S(s)$ befinden sich folglich nur die erlaubten steuerbaren und die nicht steuerbaren Ereignisse aus s. Das geschlossene Systemverhalten $L(S/G) \subseteq \mathrm{L(G)}$ mit $s \in \Sigma^*$ und $\sigma \in \Sigma$ ist nach [9] rekursiv definiert mit:

1. $\varepsilon \in L(S/G)$.
2. $(s \in L(S/G) \wedge \sigma \in S(s) \wedge s\sigma \in L(G) \Rightarrow s\sigma \in L(S/G)$.
3. keine anderen Strings gehören zu $L(S/G)$.

Der leere String ε kann niemals verboten werden und ist damit immer in $L(S/G)$ enthalten. Ein String s darf nur dann um ein Ereignis σ erweitert werden, wenn diese sowohl in S erlaubt ist und in G stattfinden kann. Die im Steuerkreis erlaubten Strings aus dem markierten ungesteuerten Systemverhalten $L_m(G)$ bilden das markierte gesteuerte Verhalten $L_m(S/G)$ mit

$$L_m(S/G) = L(S/G) \cap L_m(G). \tag{2.36}$$

Eine allgemeine Anforderung an das gesteuerte Systemverhalten ist das Nicht-Blockieren. Mit Formel (2.34) folgt daraus, dass S/G nichtblockierend ist, wenn

$$\overline{L_m(S/G)} = L(S/G). \tag{2.37}$$

Die Sprache $L(S/G)$ besteht somit nur aus den Präfixen des markierenden Verhaltens. Damit führt jede generierte Ereignisfolge immer zu einem markierten Zielzustand des Systems.

Zur Synthese des Supervisors S wird die Spezifikation K modelliert, welche das geforderte Verhalten repräsentiert. Als Entwurfsschritt muss eine **Steuerbarkeitsanalyse** von K bezüglich G durchgeführt werden. Die Steuerbarkeit ist eine Existenzbedingung für eine Steuerung. Eine Spezifikation $K \subseteq \Sigma^*$ ist steuerbar bezüglich G, wenn die Bedingung

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K} \tag{2.38}$$

erfüllt ist. Informell fordert die Gleichung, dass kein String $s \in \overline{K}$ durch das Auftreten eines nicht beobachtbaren Ereignisses $\sigma \in \Sigma_{uc}$, welches in $L(G)$ möglich ist, die Sprache von $\overline{K}$ verlassen darf. Ist die Spezifikation K nicht steuerbar, wird eine möglichst gering von K abweichende Spezifikation gesucht, die steuerbar bezüglich G ist. Diese Forderung wird durch die supremale steuerbare Teilsprache $K^{\uparrow C}$, welche etwas mehr einschränkend als K ist, erfüllt. Alternativ kann auch die infimale präfix-geschlossene steuerbare Obersprache $K^{\downarrow C}$, welche etwas mehr erlaubend als K ist, herangezogen werden. Auf eine vollständige Definition der beiden Teilsprache wird verzichtet und auf die Literatur [7], [8], [9] und [17] hingewiesen.

Die Existenz einer bezüglich G steuerbaren Spezifikation K trifft noch keine Aussage darüber, ob der synthetisierte Supervisor S nichtblockierend gegenüber G ist. Es existiert genau dann eine nichtblockierende Steuerung S bezüglich G, wenn

$$K = \overline{K} \cap L_{\mathrm{m}}(G), \tag{2.39}$$

sodass

$$L_m(S/G) = K \wedge L(S/G) = \overline{K} \tag{2.40}$$

gilt. Diese Eigenschaft wird als $L_m(G)$-Abgeschlossenheit bezeichnet. Während der Evolution der Spezifikation K darf in keinem Schritt eine Situation vorliegen, in der ein Prozess $s \in \overline{K}$ in der Strecke G als abgearbeitet markiert wird mit $s \in L_m(G)$ und dadurch die vorgegebene Spezifikation K verletzt. Ist dies der Fall, muss s in K aufgenommen werden, um eine nichtblockierende Steuerung S bezüglich G zu ermöglichen.

Neben der Steuerbarkeit, in der eine Unterteilung in steuerbare und nicht steuerbare Ereignisse erfolgt, ist auch das Kriterium der Beobachtbarkeit anhand beobachtbaren und nicht beobachtbaren Ereignisse vorhanden. Da dieses Kriterium allerdings keine Existenzbedingung darstellt und auch nicht für in dieser Arbeit durchgeführten Steuerungsentwurf relevant ist, wird auf die weiterführende Literatur [18] und [19] verwiesen.

Zur Bestimmung eines Supervisors werden verschiedene Problemfragen formuliert und deren Lösungsansätze bestimmt. In [7], [8] und [9] sind die existierenden Probleme und Lösung zusammengefasst. Für diese Arbeit ist nur das Basisproblem für nicht steuerbare Ereignisse relevant. Dieses wird als Basic Supervisory Control Problem-Nonblocking Case (BSCP-NB) beschrieben und nach [7] wie folgt definiert: Gegeben ist ein ereignisdiskretes System G mit einem Ereignisalphabet Σ mit $\Sigma_{uc} \subseteq \Sigma$ und eine $L_m(G)$-abgeschlossene Spezifikation $K \subseteq L_m$. Bestimme einen nichtblockierenden Supervisor S, sodass gilt

1. $L_m(S/G) \subseteq K$
2. $L_m(S/G)$ „maximal“ ist, also für jeden anderen nichtblockierenden Supervisor S' mit $L_m(S'/G) \subseteq K$ gilt: $L_m(S'/G) \subseteq L_m(S/G)$

Die gesuchte Steuerung muss so gewählt werden, dass

$$L(S/G) = \overline{K^{\uparrow C}}. \tag{2.41}$$

Die Repräsentation der resultierenden Steuerung S kann als Liste ihrer Steuereingriffe repräsentiert werden. Die Steuereigriffe stellen die Menge der im nächsten Schritt erlaubten Ereignisse dar. Folgende Steuereingriffe führen zu $L(S/G) = \overline{K^{\uparrow C}}$:

$$S(s) = [\Sigma_{uc} \cap \{\sigma \in \Sigma | \delta(\delta(x_0, s), \sigma)!\}] \cup \left\{\sigma \in \Sigma_c \middle| s\sigma \in \overline{K^{\uparrow C}}\right\} \tag{2.42}$$

Informell beschrieben: Die Menge $\{\sigma \in \Sigma | \delta(\delta(x_0, s), \sigma)!\}$ beinhaltet die Ereignisse aller abgehenden Transitionen aus dem Zustand $\delta(x_0, s)$ von G. Die Schnittmenge mit Σ_{uc} stellt die nicht beinflussbaren, nicht steuerbaren Ereignisse dar. In Bezug auf die Menge der steuerbaren Ereignisse müssen genau die Ereignisse $\sigma \in \Sigma_c$ in $S(s)$ enthalten sein, deren Generierung nicht zu einer Verletzung der Spezifikation $\overline{K^{\uparrow C}}$ führt.

Alternativ kann S auch als akzeptierender Generator R angegeben werden, der die Sprache $\overline{K^{\uparrow C}}$ markiert. Dafür wird $R = (Y, \Sigma, \varsigma, y_o, Y_m)$ so konstruiert, dass $Y = Y_m$, Σ identisch dem Ereignisalphabet der Strecke ist, $R = Trim(R)$ und ς, sodass $L_m(R) = L(R) \coloneqq \overline{K^{\uparrow C}}$ erfüllt ist. Unter Verwendung der strengen Synchronisation SPC (siehe Formel (2.14)) lässt sich das gesteuerte Verhalten des geschlossenen Steuerkreises wie folgt berechnen:

$$L(G||_{SPC}R) = L(G) \cap L(R) = L(G) \cap \overline{K^{\uparrow C}} = L(S/G) \tag{2.43}$$

Für das markierte gesteuerte Verhalten gilt:

$$L_m(G||_{SPC}R) = L_m(G) \cap L_m(R) = \overline{K^{\uparrow C}} \cap L(G) = L(S/G) \cap L_m(G) = L_m(S/G) \tag{2.44}$$

Die Konstruktion von R nach den beschriebenen Vorschriften und die anschließende dargestellte Produktkomposition von R und G führt exakt zum Zielverhalten des geschlossenen Steuerkreises.

2.2.7 Strukturelle Steuerungsentwurfsansätze

Die in Kapitel 2.2.6 vorgestellte Steuerungssynthese entwirft anhand eines unstrukturierten Streckenmodells den Supervisor. Dieser Entwurfsansatz wird als monolithische Entwurfsmethode bezeichnet aus der eine zentralisierte Steuerung hervorgeht, die die gesamte Steuerungsaufgabe übernimmt. Auch wenn die ungesteuerte Strecke aus einzelnen Komponenten strukturiert aufgebaut ist, führen Komposition und der monolithische Entwurfsansatz zu einem strukturlosen Gesamtmodell, welches je nach Anlagengröße eine hohe Modellkomplexität aufweisen kann. Eine hohe Modellkomplexität führt wiederum zu

längeren Berechnungszeiten der Analyse- und Entwurfsmethoden. Zusätzlich ist eine räumlich verteilte Implementierung der entworfenen Steuerung nicht möglich. Aus diesen Gründen wird der monolithische Ansatz durch Modularisierung und Hierarchisierung in seiner Entwurfsstruktur erweitert. Nachfolgend werden zwei strukturelle Ansätze der SCT vorgestellt. Eine Übersicht und Erläuterung der nicht erwähnten Entwurfsansätze sind in [7] zu finden.

Beim **modularen Steuerungsentwurf** wird die globale Steuerungsaufgabe auf mehrere Supervisor verteilt. Vorrausetzung ist, dass alle Streckenereignisse für jeden Supervisor global zur Verfügung stehen. In nachfolgender Abbildung ist ein geschlossener Steuerkreis mit zwei modularen Supervisor S_1 und S_2 dargestellt.

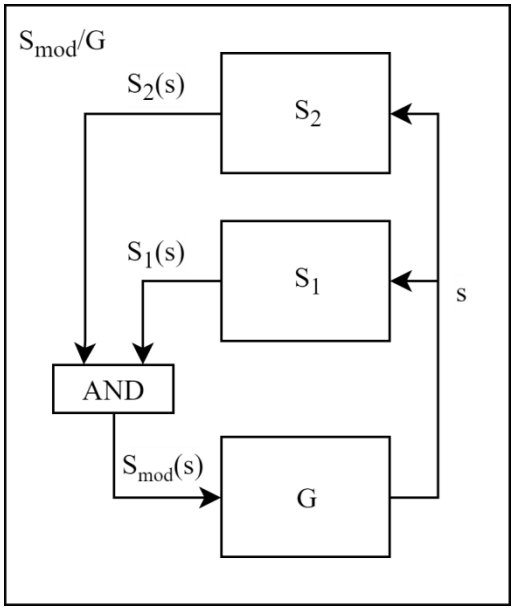

Abbildung 5: Blockschaltbild des modular geschlossenen Steuerkreises S_{mod}/G

Die Steuerungen S_1 und S_2 überwachen den von G generierten String s und erlauben jeder für sich die nach ihrer Steuerungsaufgabe erlaubten Folgeereignisse. Die Steuereingriffe werden durch $S_1(s)$ und $S_2(s)$ festgelegt. Ob ein Ereignis tatsächlich ausgeführt oder verhindert wird, legt die Boolesche Funktion *AND* fest. Somit wird ein Ereignis σ nur dann ausgeführt, wenn gilt: $\sigma \in S_1(s) \wedge \sigma \in S_2(s)$. Der allgemeine modulare Supervisor $S_{mod}: L(G) \rightarrow 2^{\Sigma}$ für n zulässige Supervisor ist nach [7] formal definiert als

$$S_{mod}(s) = S_1(s) \cap S_2(s) \cap S_3(s) \cap \ldots \cap S_n(s). \tag{2.45}$$

Aus dieser Gleichung leitet sich nach [7] und [8] das Verhalten des geschlossenen Steuerkreises S_{mod}/G wie folgt ab:

$$L(S_{mod}/G) = L(S_1/G) \cap L(S_2/G) \cap \ldots \cap L(S_n/G) \tag{2.46}$$

$$L_m(S_{mod}/G) = L_m(S_1/G) \cap L_m(S_2/G) \cap \ldots \cap L_m(S_n/G) \tag{2.47}$$

Das Modular Supervisor Control Problem (MSCP) gibt an, dass bei einem gegebenen ereignisdiskreten System mit dem Ereignisalphabet Σ, der nicht steuerbaren Ereignismenge $\Sigma_{uc} \subseteq \Sigma$ und der Spezifikation $K = K_1 \cap K_2 \cap \ldots \cap K_n$ mit $K_i = \overline{K_i}, i = [1, \ldots, n]$ die modulare Steuerung S_{mod}so bestimmt wird, dass gilt

$$L(S_{mod}/\mathrm{G}) = K^{\uparrow C}. \tag{2.48}$$

Für die modulare Lösung des Entwurfsproblems gelten die Güteanforderungen aus der monolithischen Lösung. Aufgrund der präfix-geschlossenen Teilspezifikationen lassen sich die einzelnen Steuerungen S_i unabhängig voneinander mittels

$$L(S_i/G) = K_i^{\uparrow C} \text{ für } i = 1, \ldots, n \tag{2.49}$$

entwerfen. Anschließend kann der modulare Supervisor S_{mod} unter Verwendung von Gleichung (2.45) gebildet werden. Durch die Schnittmengenbildung bleibt die Eigenschaft des Nichtblockierens nicht zwingend erhalten. Am Beispiel für zwei Steuerungen muss ein nichtblockierender modularer Supervisor nach [20] folgende Bedingung erfüllen:

$$\overline{L_m(S_1/G) \cap L_m(S_2/G)} = \overline{L_m(S_1/G)} \cap \overline{L_m(S_2/G)} \tag{2.50}$$

Der Vorteil des modularen gegenüber dem monolithischen Entwurfsansatzes ist zum einen die geringere Komplexität der einzelnen Steuerungen, zum anderen ist eine verteilte Implementierung möglich, welche flexibel hinsichtlich Änderungen ist. Es wird jedoch weiterhin ein strukturloses Gesamtmodell der Strecke benötigt. Dafür müssen trotz verteilter Hardware alle Ereignisse global verfügbar sein. Zusätzlich sind nicht alle Spezifikationen aufgrund der geforderten Steuerungsaufgabe modular zerlegbar.

Mittels des **lokal-modularen Entwurfsansatze**s kann neben der Steuerungsaufgabe auch das Streckenmodell modularisiert und strukturiert werden. Die vorgestellten Methoden sind aus [7], [21] und [22] entnommen. Die Grundidee ist, dass nur die Komponenten zusammengefasst werden, die zur Einhaltung einer bestimmten Spezifikation notwendig sind. Dies ist besonders für stark nebenläufige Komponenten zutreffend. In [21] ist ein ausführliches Anwendungsbeispiel beschrieben. In folgendem Blockschaltbild ist der geschlossene Steuerkreis S_{mod}/G dargestellt.

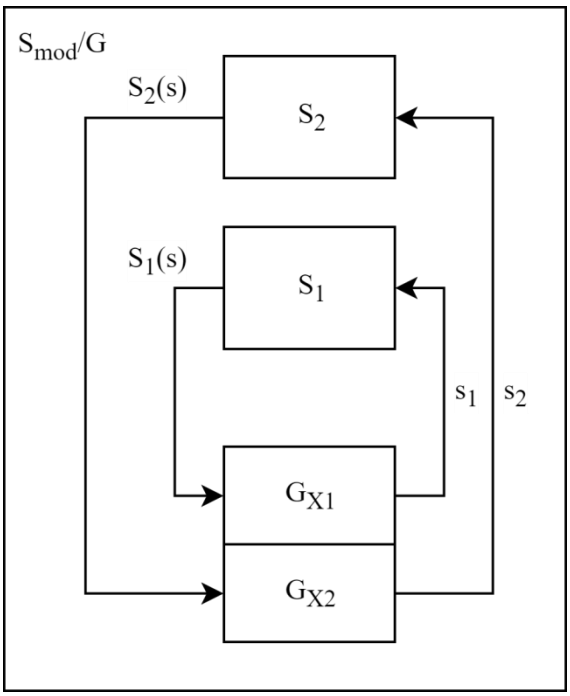

Abbildung 6: Blockschaltbild des lokal-modular geschlossenen Steuerkreises S_{mod}/G

Für die Durchführung der Streckenmodularisierung müssen zunächst die einzelnen Systemkomponenten genauer spezifiziert werden. Zur vereinfachten Darstellung wird für den parallelen Kompositionsoperator $||_{SYPC}$ auf den Index SYPC verzichtet und nur noch das Symbol $||$ verwendet. Das Kompositionssystem G aus n' Komponenten G_j' und seinem Ereignisalphabet Σ ist definiert als

$$G = ||_{i=1}^{n'} G_j' \text{ und } \Sigma = \cup_{i=1}^{n'} \Sigma_j'. \tag{2.51}$$

Sind die einzelnen Komponenten G_j' asynchron zueinander, besitzen sie keine gemeinsamen Ereignisse. Die Komposition wird dann als Produktsystem bezeichnet [16]. Das Produktsystem mit der größtmöglichen Anzahl an asynchronen Komponenten wird als feinstes Produktsystem bezeichnet. Durch die Bildung des feinsten Produktsystems wird das Streckenmodell asynchronisiert. Eine zwanghafte Synchronisation erfolgt durch die Spezifikationen, welche Ereignisse aus den einzelnen Produktsystemen enthalten. Die synchronisierten Komponenten werden als lokale Systeme bezeichnet. Gegeben sei ein Produktsystem G mit n Komponenten G_i und m Spezifikationen K_{Xj} definiert über $\Sigma_{Xj} \subseteq \Sigma$. Die lokalen Systeme G_{Xj}, welche jeweils die von einer Spezifikation K_{Xj} zwanghaft synchronisierten Komponenten G_i enthalten, sind definiert als

$$G_{Xj} = ||_{i \in N_{Xj}} G_i \text{ mit } N_{Xj} = \left\{k \in \{1, \dots, n\} \middle| \Sigma_k \cap \Sigma_{Xj} \neq \emptyset\right\}. \tag{2.52}$$

Nicht jede Komponente wird durch eine Spezifikation synchronisiert und kann einem lokalen System zugeordnet werden. Sie werden als inhärent asynchrone Komponenten bezeichnet und haben keinen Einfluss auf das Gesamtsystem. Daher können sie für den lokal-modularen Entwurf vernachlässigt werden. Das eingeschränkte Gesamtsystem G_e ergibt sich somit aus

$$G_e = ||_{i=1}^{m} G_{Xj} \text{ und } \Sigma_e = \cup_{i=1}^{m} \Sigma_{Xj}. \tag{2.53}$$

Damit die modellierten Spezifikationen K_{Xj} bezüglich der angepassten lokalen Systemen G_{Xj}, dem eingeschränkten System G_e und dem Gesamtsystem G gültig sind, müssen diese wie folgt angepasst werden:

$$E_{Xj} = K_{Xj} || L_m(G_{Xj}) \tag{2.54}$$

$$E_{Xj_e} = K_{Xj} || L_m(G_e) \tag{2.55}$$

$$E_X = K_{Xj} || L_m(G) \tag{2.56}$$

Zum besseren Verständnis der dargestellten Modellanpassungen, soll folgendes Beispiel aus [7] herangezogen werden. Gegeben sei das aus fünf Komponenten G_i' mit $i \in N' = \{1, \dots, 5\}$ bestehende Kompositionssystem G. Die geforderte Steuerungsaufgabe ist in den gegebenen Spezifikationen $K_{X1} \subset \Sigma_{X1}^*$ und $K_{X2} \subset \Sigma_{X2}^*$ enthalten. In folgender Abbildung werden die Relationen zwischen den Ereignisalphabeten in Form eines Venn-Diagramms dargestellt.

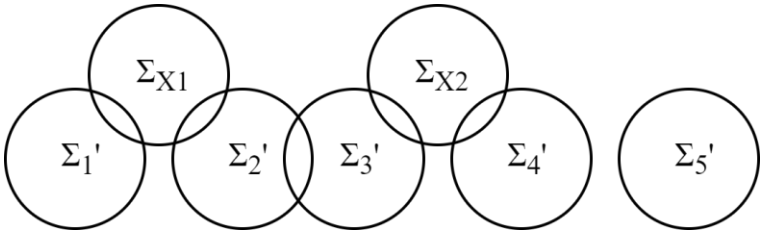

Abbildung 7: Venn-Diagramm der Ereignisalphabete [7]

Das feinste Produktsystem ergibt sich aus dem in Abbildung 7 dargestellten Venn-Digramm mit $G_1 = G_1'$, $G_2 = G_2' || G_3'$, $G_3 = G_4'$ und $G_4 = G_5'$. Anhand der zwanghaften Synchronisation durch die gegebenen Spezifikationen ergeben sich die lokalen Systeme mit $G_{X1} = G_1 || G_2$ und $G_{X2} = G_2 || G_3$. Die Komponente G_4 ist in diesem Fall inhärent asynchron und keinem lokalen System zugeordnet. Daraus folgt das eingeschränkte System $G_e = G_1 || G_2 || G_3$. Die Spezifikationen werden anschließend durch $E_{X1} = K_{X1} || G_{X1}$ und $E_{X2} = K_{X2} || G_{X2}$ angepasst. Für die Anpassung bezüglich des eingeschränkten Systems und des Gesamtsystems gilt Entsprechendes [7].

Für den abschließenden lokal-modularen Steuerungsentwurf wird eine lokale Modularität gefordert. Diese ist durch die formale Definition für lokale Modularität von Sprachen nach [7]wie folgt definiert*: l sei eine beliebige Indexmenge und $L_i \subseteq \Sigma_i^*$, $i \in l$. Somit ist die Menge $\{L_i, i \in l\}$ lokal-modular, wenn*

$$||_{i \in l} \overline{L_i} = \overline{||_{\iota \in l} L_\iota}. \tag{2.57}$$

Es gilt für zwei lokale Supervisor nach [22] *folgender Zusammenhang: Für ein gegebenes Kompositionssystem G mit n' Komponenten und zwei lokalen Spezikationen K_{X1} und K_{X2} sei $supC(E_{X1}, G_{X1})$ die supremale steuerbare Teilsprache $E_{X1}^{\uparrow C}$ von E_{X1} bezüglich G_{X1} und $supC(E_{X2}, G_{X2})$ die supremale steuerbare Teilsprache $E_{X2}^{\uparrow C}$ von E_{X2} bezüglich G_{X2}. Sind die supremal steuerbaren Teilsprachen lokal-modular, dann gilt*

$$supC(E_{X1e} \cap E_{X2e}, G_e) = supC(E_{X1}, G_{X1}) || supC(E_{X2}, G_{X2}) \tag{2.58}$$

Gleichung (2.58) sagt aus, dass es ausreichend ist die gegebenen Spezifikationen K_{X1} und K_{X2} bezüglich ihrer lokalen Systeme auszudrücken und die Supervisor mittels $L(S_1/G_{X1}) = supC(E_{X1}, G_{X1})$ und $L(S_2/G_{X2}) = supC(E_{X2}, G_{X2})$ zu bestimmen. Abschließend können die Steuerungen nach [23] um ihre redundanten Zustände reduziert werden.

Neben den modularen und lokalen-modularen Entwurfsansatz, gibt es noch den dezentralen Ansatz. Dieser zeichnet sich durch Berücksichtigung partieller Beobachtbarkeit aus. Der dezentrale Ansatz wird für diese Arbeit nicht angewendet und daher nicht weiter beschrieben. Es wird daher auf die Literatur [24] und [25]verwiesen.

2.2.8 Modellierungsprozess

Die vorgestellten Methoden und Ansätze werden zur Durchführung eines modellbasierten Steuerungsentwurfs auf Basis ereignisdiskreter Systeme herangezogen. Im Gegensatz zum modellbasierten Entwurfsansatz steht der informelle intuitive Ansatz, bei welchem der Anwender durch auf Basis eines Konzeptentwurfs den geforderten Steuerungscode erstellt und implementiert. Eine Verifikation der Software findet ebenfalls überwiegend informell statt. Im Falle einer Falsifikation wird die Software rekursiv nachgebessert und erneut getestet.

In dieser Arbeit wird der modellbasierte Ansatz gewählt. In folgender Abbildung sind die einzelnen Schritte von der Anforderungsanalyse bis zur Softwareimplementierung dargestellt.

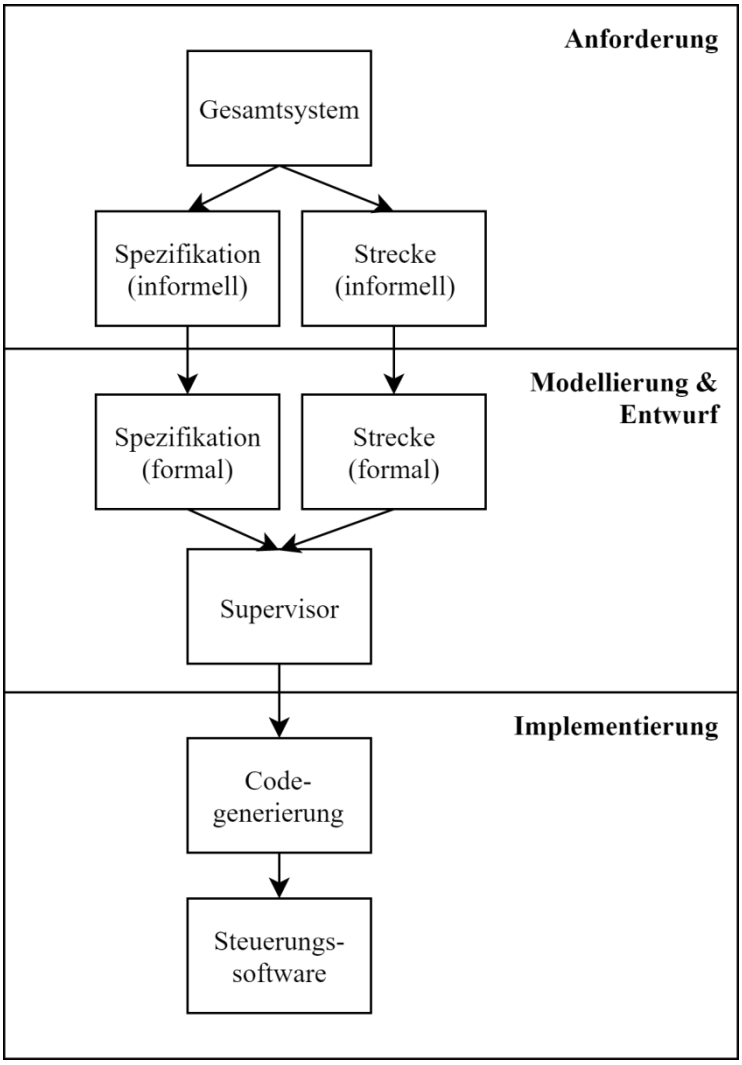

Abbildung 8: Entwicklungsschritte eines modellbasierten ereignisdiskreten Steuerungsentwurfs (angelehnt an [6])

Am Anfang des Modellierungsprozess steht das Verständnis des realen Gesamtsystems, welches in seinem Verhalten gesteuert werden soll. Durch eine idealisierte Betrachtung sowie Abstraktion und Strukturierung wird eine informelle Beschreibung des Gesamtsystems durchgeführt. Dabei wird zwischen dem zu steuernden System (Strecke) und der Steuerungsaufgabe (Spezifikation) unterschieden. In der informellen Betrachtung ist der Fokus auf die relevanten Prozessparameter zu legen, welche sich aus der geforderten Aufgabe ableiten [26].

Anschließend wird unter Nutzung der vorgestellten mathematischen Beschreibungsformen ein formales Modell erstellt. Die Unterteilung in Strecke und Spezifikation bleibt erhalten. Die Qualität des berechneten Supervisors ist maßgeblich vom erstellten formalen Modell beider

Komponenten abhängig. Mittels Analyse- und Simulationsmethoden, kann nun die synthetisierte Steuerung verifiziert werden.

Durch einen geeigneten Codegenerator kann die entworfene Steuerung in einen ablaufbaren Softwarecode umgewandelt werden. Die Codegenerierung hängt von der eingesetzten Entwicklungssoftware ab, welche für die Modellbildung und den Steuerungsentwurf genutzt wird, sowie von der Laufzeitumgebung auf welcher der generierte Steuerungscode ablaufen soll.

2.3 Dezentrale Energiesysteme

Bei dezentralen Energiesystemen wird elektrische Energie verbrauchernah erzeugt und verteilt. Die Leistungsfähigkeit ist meist auf die Deckung des lokalen Stromverbrauchs ausgelegt. Im Gegensatz zur zentralen Energieversorgung wird der erzeugte Strom nicht über ein zentrales Hochspannungsnetz, welches je nach gegebener Infrastruktur Länder- oder Kontinente miteinander koppelt, übertragen, sondern über ein lokales Verteilnetz [27].

Die Erzeugerkomponenten in einem dezentralen Energiesystem bestehen überwiegend aus regenerativen Erzeugern. Die bekanntesten und am weitesten verbreiteten sind Photovoltaik- und Windkraftanlagen. Dennoch können auch nicht regenerative Quellen als Energieerzeuger verwendet werden.

Die Integration von Speicherkomponenten stellt ein weiteres Unterscheidungsmerkmal zu zentralen Netzen dar. Diese werden genutzt, um die fluktuierende Energieerzeugung zu kompensieren oder aber auch um hohe Lastspitzen im Verbrauch zu decken. Die Speichertechnologien unterscheiden sich in ihrer Energiedichte, sowie der Leistungsaufnahme und -abgabe.

Das Verteilnetz (auf Englisch: Grid) koppelt die angeschlossenen Erzeuger-, Last und Speicherkomponenten miteinander. Es kann als Wechselstromnetz (auf Englisch: Alternating current (AC)-Grid) oder Gleichstromnetz (auf Englisch: Direct current (DC)-Grid) ausgelegt sein. Der Vorteil im AC-Netz liegt in der Kurzschlusserkennung über den Nulldurchgang. Ein Nachteil sind die höheren Übertragungsverluste im Vergleich zum DC-Netz. Das DC-Netz eignet sich hingegen bei kleinskalierten Netzstrukturen, da die hier als Erzeuger meist Photovoltaikanlagen genutzt werden, welche bereits eine Gleichspannung besitzen und damit nicht erst in Wechselstrom umgewandelt werden müssen. Gleiches gilt für kleine Lasten, wie Elektronikgeräte oder Leuchtdioden. Dadurch werden die Wandlungsverluste verkleinert und damit die Effizienz des Gesamtsystems gesteigert.

Um die einzelnen Erzeuger-, Speicher- und Lastkomponenten an das Verteilnetz zu koppeln, werden leistungselektronische Einheiten in Form von DC/DC-Konvertern oder AC/DC Konvertern als zentrale Stellglieder verwendet. Diese wandeln eine am Eingang zugeführte Gleich- oder Wechselspannung in eine höhere oder niedrigere Ausgangsspannung um. Eine detaillierte Beschreibung von DC/DC und AC/DC Konvertern kann in [28] eingesehen werden.

Häufig wird im Kontext dezentraler Energiesysteme der Begriff des Smart-Grids verwendet, welcher allgemein ein intelligentes Energienetz beschreibt. Der Einsatz von Kommunikationstechnik, lässt eine intelligente Regelung und Steuerung von einzelnen Systemkomponenten zu, um zum einen dynamisch auf Strom- und Spannungsschwankungen im Verteilnetz reagieren zu können und zum anderen einen Datenaustausch zwischen einzelnen Energiesystemen aufzubauen [27].

Der Begriff des Nano-Grids beschreibt im Kontext der dezentralen Energiesysteme ein kleinskaliertes dezentrales Verteilnetz. Die Leistung dient dabei als Kenngröße. Die absolute Leistungsgrenze wann ein Verteilnetz als Nano-Grid bezeichnet wird, ist in der Literatur nicht fest vorgegebenen. Als Orientierung soll der Leistungsbedarf von Privathaushalten herangezogen werden. Damit kann ein Energienetz sowohl in einem Ein-Familien-Haushalt, als auch in einem Mehrfamilienhaus als Nano-Grid bezeichnet werden. Die Kopplung mehrerer Nano-Grid-Systeme und damit die Skalierung der Leistung und Kapazität, führen zum Begriff des Micro-Grids, welches eine nächsthöhere Leistungsklasse von dezentralen Energiesystemen beschreibt [29].

2.4 Entwicklungsumgebung DESTool

Als Entwicklungswerkzeug wird in dieser Arbeit die Software DESTool genutzt. DESTool wurde am Lehrstuhl für Regelungstechnik der Friedrich-Alexander Universität Erlangen-Nürnberg entwickelt. Es ist unter der freien Open Source GNU Lesser General Public Lizenz (LGPL) verfügbar. DESTool stellt eine grafische Benutzerschnittstelle zur implementierten *libFAUDES* Bibliothek dar. Diese enthält die benötigten Algorithmen zur Handhabung von DFA und regulären Sprachen. Neben der grafischen Modellierung von Automaten, können damit auch die Analyse- und Entwurfsfunktionen der SCT eingesetzt werden. In folgende Abbildung ist das Hauptmenü dargestellt. Es werden nachfolgend nur die verwendeten Funktionen der Software dargestellt. Eine vollständige Erläuterung ist in [30] zu finden.

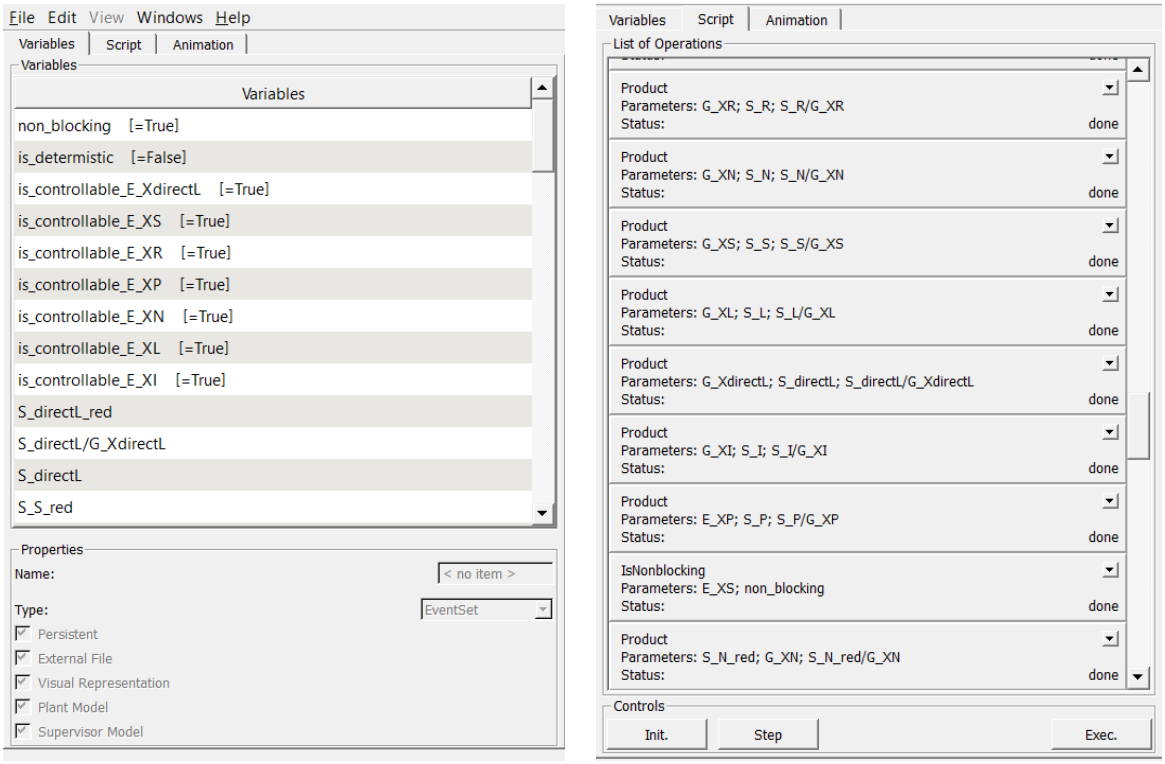

Abbildung 9: Hauptmenü DESTool mit geöffnetem Projekt (links: Reiter *Variables*, rechts: Reiter *Script*)

Im Reiter *Variables* werden Variablen deklariert. Dazu stehen verschiedene Datentypen zur Verfügung. Die in dieser Arbeit genutzten Typen schränken sich auf *System*, *Boolean* und *Alphabet* ein. Im Datentyp *System* werden die Generatoren für Strecke und Spezifikation erstellt. Der Datentyp *Boolean* dient als Zielvariable für den logischen Rückgabewert einer Funktion, wie zum Beispiel *isDeterministic*, die einen Generator auf deterministisches Verhalten prüft. In der Variablen *Alphabet* kann das Ereignisalphabet eines Generators separat abgespeichert werden.

Im Reiter *Script* können die verschiedenen Funktionen für Komposition, Analyse und Synthese genutzt werden. Dabei können alle Funktionen sequentiell auf einmal berechnet werden oder Schrittweise. Die für diese Arbeit relevanten Funktionen sind

Tabelle 1: Übersicht der genutzten DESTool Funktionen

Funktionsname	**Beschreibung**
$Parallel(G_1, G_2, G_3)$	Führt eine parallele Komposition der Generatoren G_1 und G_2 durch und speichert das Ergebnis in G_3.
$Product(G_1, G_2, G_3)$	Führt eine Produktkomposition der Generatoren G_1 und G_2 durch und speichert das Ergebnis in G_3.
$AlphabetExtract(G, A)$	Extrahiert das Ereignisalphabet aus G und speichert es in A.
$SelfLoop(G, A)$	Fügt eine Schlinge an jedem Zustand von G mit den Ereignissen von A.

$isControllable(G, K, res)$	Prüft K auf Steuerbarkeit bezüglich G und schreibt Ergebnis in res.
$isNonBlocking(G, res)$	Prüft G auf nichtblockieren und schreibt Ergebnis in res.
$SupConNB(G, K, S)$	Berechnet die nichtblockierende supremale steuerbare Teilsprache auf Basis G und K und speichert das Ergebnis als Generatorsystem in S.
$SupReduce(S, G, S_{red})$	Berechnet den reduzierten Supervisor von S bezüglich G und speichert das Ergebnis in S_{red}.

Um das Ergebnis eines ereignisdiskreten Steuerungsentwurfes interaktiv zu prüfen, bietet DESTool im Reiter *Animation* eine schrittweise Ansteuerung von Ereignissen an. Nach jeder Ereignisansteuerung sind der jeweilige Zustand der Komponenten erkennbar sowie das erlaubte und das nicht erlaubte Ereignisalphabet. Diese Ansicht wird für die Simulation der entworfenen Steuerungen genutzt.

3.Beschreibung des Nano-Grid-Modells

In diesem Kapitel wird das entworfene Nano-Grid Systemmodell beschrieben. Der Fokus liegt auf der theoretischen Modellbetrachtung. Die in diesem Kapitel beschriebenen Modellgrundlagen sind aus [1], [31], [32], [33] und [34] zusammengeführt. Es werden die in diesen Publikationen entwickelten Regelungskonzepte für eine stabile Regelung von Strom und Spannung herangezogen und für einen ereignisdiskreten Entwurf angepasst. Die Systembeschreibung wird soweit abstrahiert, dass es für den durchgeführten Steuerungsentwurf ausreichend ist.

Neben dem allgemeinen Überblick über die einzelnen Systemkomponenten, werden die Betriebszustände der eingesetzten Leistungselektronikmodule beschrieben. Anschließend werden die jeweiligen Regel- und Steuerungskonzepte näher erläutert, dabei liegt der Fokus auf der Steuerungsebene. Abschließend werden drei verschiedene Steuerungsarchitekturen gegenübergestellt. Das betrachtete Model dient als Grundlage für den ereignisdiskreten Steuerungsentwurf.

3.1 Funktion des Nano-Grid Systems

Die Hauptaufgabe ist das Lastmanagement durch priorisiertes Zu- und Abschalten der Lasten, Erzeuger und Speichermodule. Das geforderte Ziel ist den Anteil an regenerativ erzeugtem Strom so effizient wie möglich zu nutzen und gleichzeitig die Lasten so stabil wie möglich zu versorgen. Darüber hinaus wird der Anlaufprozess bei Unterbrechung durch ein globales Fehlerereignis gesteuert. Eine weitere Funktion ist das Verhindern der Überladung und Tiefenentladung der angeschlossenen Speicherkomponente. Im Hinblick auf diese Steuerungsaufgaben wird das DC-Nano-Grid, nachfolgend als Nano-Grid bezeichnet, konzipiert und modelliert. Das System soll durch ein möglichst allgemeingültiges Modell abgebildet werden, um die entworfene Steuerung für verschiedene Nano-Grid Konzepte einsetzen zu können.

3.2 Beschreibung der Systemkomponenten

Im vorliegenden Modell sind regenerative als auch nicht regenerative Erzeugerkomponenten, eine Speicherkomponente und zwei Lasten vorhanden. Zusätzlich werden zwei Kopplungskomponenten integriert. Diese bilden eine Schnittstelle zu weiteren Nano-Grid Systemen und zum öffentlichen Energienetz. Aufgrund der Integration von nicht regenerativen Erzeugerquellen wird das Nano-Grid auch als hybrides Nano-Grid bezeichnet. Das Nano-Grid ist so ausgelegt, dass der Energieaustausch zwischen Erzeuger, Speicher und Verbraucher möglichst autark stattfindet. Um eine Skalierung der Leistung und Speicherkapazität zu erreichen ist optional eine Schnittstellte zu weiteren Nano-Grid Systemen vorgesehen. Weiterhin ist eine Anbindung an das öffentliche Netz möglich um Leistungsengpässe abzudecken, welche nicht im Verbund bereitgestellt werden können.

Als Komponenten werden leistungselektronische Stellglieder betrachtet, welche die Spannung der Erzeuger-, Speicher- und Lastsysteme auf die Spannung des Verteilnetzes anpassen und diese voneinander entkoppeln. Bei der Leistungselektronik handelt es sich um DC/DC-Konverter (auch DC/DC-Wandler oder Gleichspannungswandler genannt). Über eine Pulsweitenmodulation (PWM) werden die Leistungstreiber in den Gleichspannungswandler angesteuert. Diese sind sowohl als Tief- als auch Hochsetzsteller ausgelegt. Die Hauptparameter des Systems sind Spannung und Strom im Verteilnetz. Diese müssen geregelt und überwacht werden damit bestimmte Wertgrenzen eingehalten werden. Die Folge bei Über- oder Unterschreiten von definierten Grenzwerten sind Hardwareschäden der angeschlossenen, da diese nur in einem bestimmten Wertebereichen fehlerfrei funktionieren.

In nachfolgender Abbildung ist das Gesamtsystem DC-Nano-Grid mit den leistungselektronischen Systemkomponenten abgebildet.

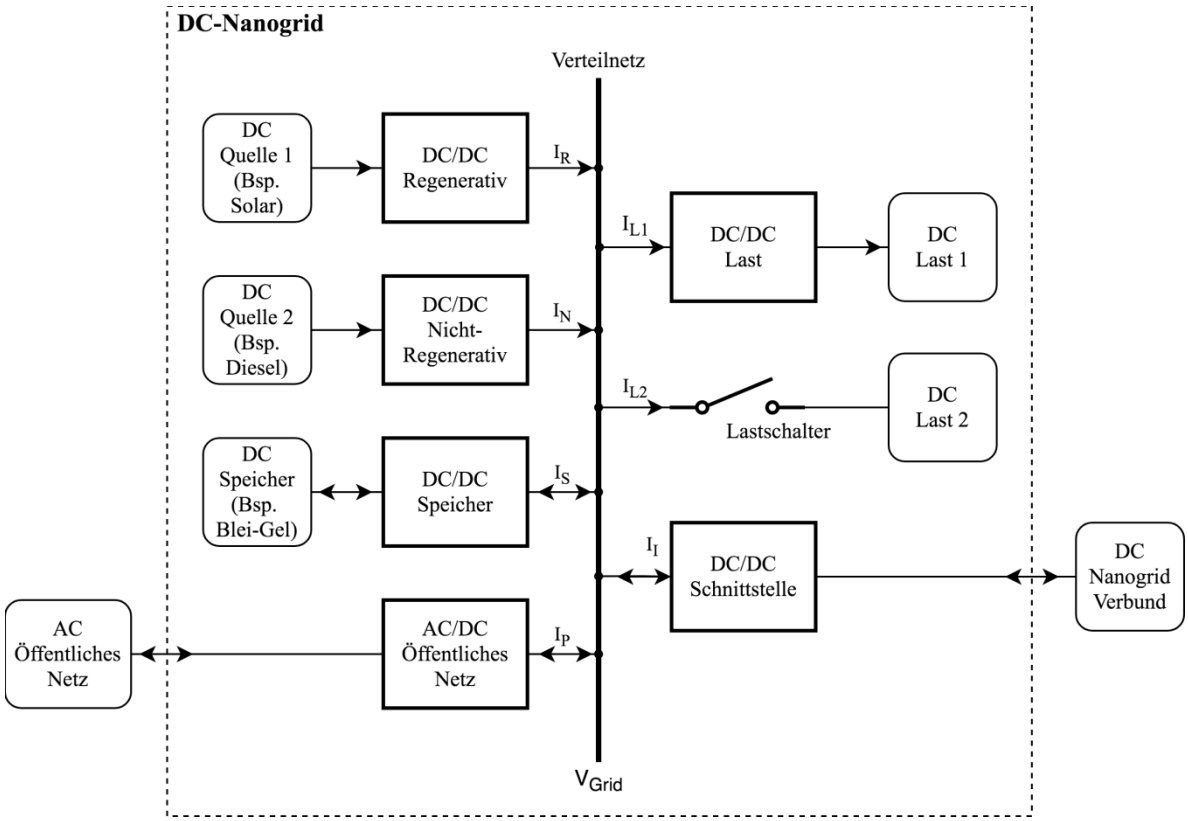

Abbildung 10: Systemübersicht des zu steuernden DC-Nano-Grids

Zentral ist das Verteilnetz abgebildet, welches als Zweidrahtleitung ausgelegt ist und alle angeschlossenen Komponenten miteinander verbindet. Die Netzspannung V_{Grid} dient als Referenzspannung nach der die angeschlossenen Module ausgelegt werden. Für kleine DC Netze wie es das Nano-Grid darstellt, werden typischerweise Spannungspotentiale von 12 V bis 60 V genutzt. Bei hohen Leistungen sind auch Spannungen bis 720 V möglich. Der tatsächliche absolute Wert ist für die Modellbetrachtung dieser Arbeit irrelevant.

Der Stromfluss der einzelnen Komponenten ist durch Pfeile dargestellt. Hier ist ein Einspeisen in das Verteilnetz mit positivem Strom möglich, sowie ein Ausspeisen aus dem Verteilnetz mit negativem, Strom. Erzeuger speisen in das Netz, Lasten speisen aus dem Netz, Speicher und Schnittstellenkomponenten führen beide Funktionen durch.

Alle Komponenten besitzen ein global definiertes Fehlerereignis, welches in der Realität durch das Betätigen eines Not-Halt Schalters ausgelöst werden kann. Dieses Ereignis lässt alle Komponenten in einen definierten Fehlerzustand laufen.

3.2.1 Erzeugerkomponenten

Die regenerative Erzeugerkomponente *DC/DC Regenerativ* ist für die Stromerzeugung aus einer erneuerbaren Energiequelle verantwortlich. Dabei muss neben der Spannungsanpassung der angeschlossenen Quelle auch der optimale Energieertrag erzielt werden. Die Eingangsseite kann zum Beispiel eine Photovoltaikanlage oder eine Kleinwindanlage darstellen. Die Komponente *DC/DC Regenerativ* regelt den Stromfluss zwischen dem Erzeugersystem und dem Verteilnetz. Die für diese Arbeit relevanten Parameter sind der Ausgangsstrom I_R (Index R für Regenerativ) zum Verteilnetz hin und die resultierende Leistung P_R mit $P_R = V_{Grid} \cdot I_R$.

Neben der regenerativen Quelle kann auch ein nicht regeneratives Erzeugersystem angeschlossen werden, zum Beispiel ein Dieselgenerator. Das nicht regenerative Erzeugersystem ist für die Versorgung von hohen Leistungsspitzen vorgesehen, welche nicht durch Abschalten von Lasten verhindert werden können. Ein Szenario sind kritische Lasten wie Kühlaggregate, welche bei hoher Temperatur einen erhöhten Leistungsbezug aufweisen. Zum Anschluss von nicht regenerativen Erzeugern wird die Komponente *DC/DC Nicht-Regenerativ* eingesetzt. Die relevanten Parameter sind der Ausgangsstrom I_N (Index N für Nicht-Regenerativ) und die resultierende Leistung P_N mit $P_N = V_{Grid} \cdot I_N$. Für den abstrahierten Entwurf ist auch eine Kopplung eines Wechselstromgenerators an das Verteilnetz möglich. Hierzu muss die Leistungselektronik eingangsseitig zur Quelle hin an ein Wechselstromnetz angepasst werden. Die Ausgangsseite zum Verteilnetz bleibt in ihrer Steuerung identisch.

Die Komponenten *DC/DC Regenerativ* und *DC/DC Nicht-Regenerativ* besitzen vier Betriebszustände, die ihrer Funktionsweise gleich sind und daher zusammenfassend beschrieben werden.

Tabelle 2: Beschreibung der Betriebszustände der Erzeugerkomponenten *DC/DC (Nicht-) Regenerativ*

Betriebszustand	**Beschreibung**
Idle	Komponente ist betriebsbereit. Es ist keine Regelung aktiv. Es findet kein Energietransfer statt.
Constant Power (CP)	Komponente regelt auf optimalen Betriebspunkt der Erzeugerquelle. Es wird die maximale Leistung abgegeben
Constant Voltage (CV)	Komponente regelt auf vorgegebene Spannung. Voltage-Droop Regelung ist aktiv. Es wird die benötigte Spannung abgegeben
Error	Komponenten befinden sich im Sicherheitszustand

Im Zustand *Idle* ist keine Regelung aktiv. Die Hardware ist eingeschaltet und Signale können gelesen und verarbeiten werden. Dieser Zustand kann auch als Handbetrieb genutzt werden um manuell Parameter anzupassen. Im Zustand *Constant Power (CP)* regelt die Erzeugerkomponente auf den optimalen Betriebspunkt um die maximale Leistung an das Verteilnetz abzugeben. Bei regenerativen Quellen wird diese Regelung üblicherweise als Maximum Power Point Tracking (MPPT) bezeichnet. Der Zustand *Constant Voltage (CV)* aktiviert die Voltage-Droop Regelung, in welcher die Ausgangsspannung zum Verteilnetz hin geregelt wird. Die Regelstrategie wird in Kapitel näher erläutert. Der Zustand *Error* wird bei Eintreten eines Fehlereignissen aktiviert und führt die Komponente in einen Sicherheitszustand über.

3.2.2 Speicherkomponente

Die Speicherkomponente *DC/DC Speicher* ist für das Laden und Entladen des angeschlossenen Speichermediums, sowie für die Spannungsanpassung zwischen Speicher und Verteilnetz zuständig. Die verwendete Speichertechnologie ist für den Steuerungsentwurf irrelevant, da nur die Ausgangseite zum Verteilnetz betrachtet wird. Der Ladungsaustausch erfolgt bidirektional. Es wird sowohl Leistung aus der Batterie in das Verteilnetz gespeist als auch aus dem Verteilnetz in die Batterie ausgespeist. Die relevanten Parameter sind der Strom I_S (Index *S* für Speicher) und die Leistung P_S mit $P_S = V_{Grid} \cdot I_S$. Der Strom ist bei Laden der Batterie negativ und bei Entladen der Batterie positiv.

Zusätzlich ist auch die Batteriekapazität für den Steuerungsentwurf relevant. Dieser wird in Form des State of Charge (SOC) abgebildet, welcher relativ zur Nominalkapazität der Batterie angegeben wird. Die Komponente *DC/DC Speicher* besitzt damit folgende Betriebszustände.

Tabelle 3: Beschreibung der Betriebszustände der Speicherkomponente *DC/DC Speicher*

Betriebszustand	**Beschreibung**
Idle	Komponente ist betriebsbereit. Es findet kein Energietransfer statt.
Charge (Chrg)	Batterie wird geladen. Es wird Leistung aus dem Verteilnetz gezogen mit $I_S < 0$
Discharge (Dischrg)	Batterie wird entladen. Es wird Leistung in das Verteilnetz gespeist mit $I_S > 0$
Error	Komponente befindet sich im Sicherheitszustand

Alternativ zur Anbindung der Batterie anhand eines DC/DC-Konverters, kann eine Batterie auch direkt an das Verteilnetz angeschlossen werden. Dies führt dazu, dass die Netzspannung V_{Grid} direkt von der Batteriespannung abhängt. Dieser Fall wird in dieser Arbeit nicht weiter betrachtet.

3.2.3 Lastkomponenten

Die Anschaltung der zwei dargestellten Lasten erfolgt zum einen mit einem DC/DC-Konverter zum anderen direkt über einen einfachen Lastschalter. Über den Gleichspannungswandler *DC/DC Last* können Lasten mit einer unterschiedlichen Spannung als das Verteilnetz angeschlossen werden. Zusätzlich ist ein Stellglied zur Vorgabe der abgegebenen Leistung an die Last verfügbar. Dieses wird bei der Realisierung des Lastmanagement benötigt. Die relevanten Parameter für *DC/DC Last* sind der Strom I_{L1} und die Leistung P_{L1} mit $P_{L1} = V_{Grid} \cdot I_{L1}$. Die Komponente *DC/DC Last* besitzt folgende Betriebszustände.

Tabelle 4: Beschreibung der Betriebszustände der Lastkomponente *DC/DC Last*

Betriebszustand	Beschreibung
Off	Komponente ist betriebsbereit. Es findet kein Energietransfer statt. Es wird keine Leistung an die Last abgegeben.
On	Es wird die maximal benötigte Leistung an die Last abgegeben
Limit	Es wird eine begrenzte Leistung an die Last abgegeben.
Error	Komponente befindet sich im Sicherheitszustand.

Im Zustand *Limit* wird die maximal verfügbare Leistung im Netz an die Last abgeben. Somit wird der Ausgangsstrom I_{L1} begrenzt. Im Zustand *Off* ist die Hardware weiterhin aktiv analog zum Zustand *Idle* der vorher beschriebenen Komponenten, allerdings wird die Leistungsversorgung zur Last hin unterbrochen. Im Zustand *On* wird der Last die maximal benötigte Leistung bereitgestellt. Für den abstrahierten Steuerungsentwurf kann auch eine Wechselstromlast angeschlossen werden. Hierzu muss die Leistungselektronik ausgangsseitig zur Last hin angepasst werden. Zusätzlich soll erwähnt werden, dass Lasten auch direkt ohne Schalter angeschlossen werden können. Diese Variante wird allerdings betrachtet, da kein Stellglied zur Ansteuerung zur Verfügung stehen würde.

Die über einen Lastschalter angeschlossene Last dient als wirtschaftlich günstigere Alternative, da keine Leistungselektronischen Bauteile benötigt werden. Hier wird der Verbraucher logisch zu- oder abgeschaltet. Das digitale Steuersignal wird je nach Steuerungsarchitektur von einer zentralen Steuerung oder von einem der DC/DC-Konverter gesetzt. Es wird angenommen, dass die Konverter Digitale Ausgänge besitzen. Die angeschlossene Last muss für den Spannungswert V_{Grid} des Verteilnetzes ausgelegt sein. Die Betriebszustände sind auf *Off* und *On* beschränkt. Die relevanten Parameter für den Lastschalter sind der Strom I_{L2} und die Leistung P_{L2} mit $P_{L2} = V_{Grid} \cdot I_{L2}$. Für den Steuerungsentwurf werden die Lastströme zu I_L zusammengefasst mit $I_L = I_{L1} + I_{L2}$. Für die Leistung gilt Entsprechendes.

3.2.4 Schnittstellenkomponenten

Die Komponenten *DC/DC Schnittstelle* und *AC/DC Öffentliches Netz* bilden eine Schnittstelle des Nano-Grids an externe Energiesysteme. Sie können optional integriert werden um zum einen den lokalen Energiebedarfs sicherzustellen und zum anderen eine Skalierung der Systemleistung und -kapazität zu ermöglichen.

Mit Hilfe der Komponenten *DC/DC Schnittstelle* ist ein Energietransfer zwischen mehreren Nano-Grids möglich, die gemeinsam ein neues Gesamtsystem bilden, welches je nach Größe auch als Micro-Grid bezeichnet werden kann. Eine Kopplung kann durch zwei unterschiedliche Topologien folgen. Zum einen ist eine Reihenschaltung von einzelnen Nano-Grid Systemen möglich, wie in folgender Abbildung dargestellt.

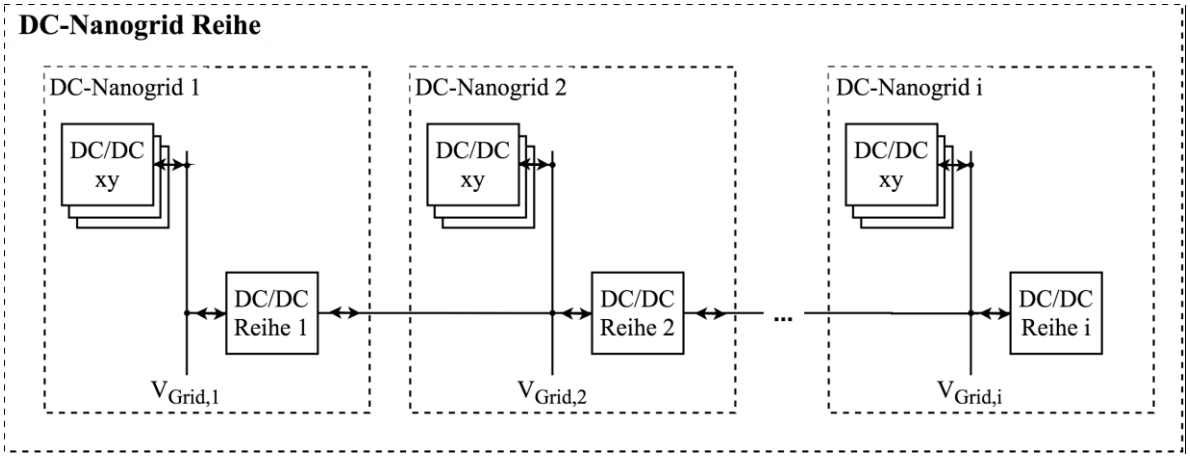

Abbildung 11: Systemübersicht von gekoppelten Nano-Grids in Reihe

Jedes Nano-Grid besitzt sein lokales Verteilnetz sowie die jeweiligen DC/DC-Konverter für Erzeuger, Speicher und Lasten (diese sind in Abbildung 11 zusammengefasst als *DC/DC xy* bezeichnet). Der bidirektionale Energietransfer zwischen den gekoppelten Nano-Grids erfolgt über die Schnittstellenkomponente *DC/DC Reihe*. Diese ermöglicht zusätzlich eine Spannungsanpassung der angeschlossenen Nano-Grids. Damit kann jedes Nano-Grid eine lokale Referenzspannung $V_{Grid,i}$ besitzen. Jede Schnittstellenkomponente hat die Informationen über den Zustand des eigenen Nano-Grids und die des benachbarten angeschlossenen Nano-Grids. Somit kann der Spannungswandler *DC/DC Reihe 1* von den Ladungsaustausch zwischen DC-Nano-Grid 1 und 2 steuern, allerdings nicht den Ladungsaustausch zwischen DC-Nano-Grid 1 und 3. Vorteil dieser Art der Verschaltung ist die einfache Skalierung durch das Zusammenschalten zweier Systeme ohne zusätzliches Verteilnetz. Es wird lediglich eine Leitung zwischen den einzelnen Nano-Grids benötigt. Nachteil ist, dass der Energieaustausch nur zwischen zwei benachbarten Nano-Grids erfolgen kann.

Neben der Zusammenschaltung einzelner Nano-Grids in Reihe, ist auch eine Kopplung durch ein zentrales Verbundnetz möglich, wie in folgender Abbildung zu erkennen ist.

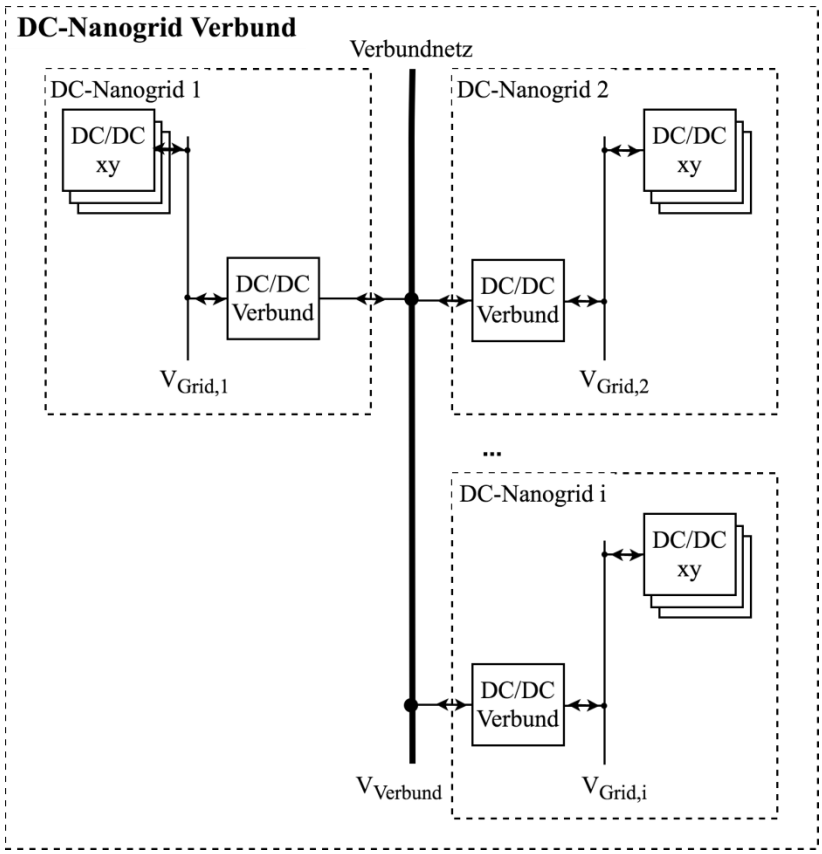

Abbildung 12: Systemübersicht von gekoppelten Nano-Grids im Verbund

An einem zentralen Verbundnetz mit der Referenzspannung $V_{Verbund}$ werden einzelne Nano-Grids angeschlossen. Die Kopplung erfolgt ebenfalls durch einen bidirektionalen Gleichspannungswandler *DC/DC Verbund*, welcher analog zu *DC/DC Reihe* sowohl eine Spannungsanpassung als auch einen Ladungsaustausch zwischen dem lokalen Netz und dem zentralen Verbundnetz durchführt.

Das Verbundnetz fungiert als zentrale Schnittstelle zwischen allen angeschlossenen Nano-Grids. Damit kann ein Ladungsaustausch zwischen allen Nano-Grids erfolgen. Die Spannung des Verbundnetzes kann um ein vielfaches höher gesetzt werden, was zu geringeren Verlusten bei der Übertragung führt und somit eine größere räumliche Distanz zwischen den Nano-Grids ermöglicht. Die Betriebszustände der Komponente *DC/DC Schnittstelle* sind für beide Topologien gleich.

Tabelle 5: Beschreibung der Betriebszustände der Schnittstellenkomponente *DC/DC Schnittstelle*

Betriebszustand	Beschreibung
Idle	Komponente ist betriebsbereit. Es findet kein Energietransfer statt.
Power internal	Leistung wird von extern in das lokale Nano-Grid übertragen.
Power external	Leistung wird vom lokalen Nano-Grid nach extern übertragen.
Error	Komponente befindet sich im Sicherheitszustand.

Der Zustand *Power internal* führt zu einem Energietransfer vom angeschlossenen externen System, welche je nach Topologie ein weiteres Nano-Grid oder das zentrale Verbundnetz, zum lokalen Nano-Grid. Der Zustand *Power external* führt zu einem Energietransfer vom lokalen Nano-Grid zum angeschlossenen externen System.

Neben der Kopplung mit einem weiteren Nano-Grid-System kann auch eine Kopplung zum öffentlichen Energienetz stattfinden. Hierfür steht die Komponente *AC/DC Öffentliches Netz* zur Verfügung. Die Betriebszustände der Komponente sind wie folgt beschrieben.

Tabelle 6: Beschreibung der Betriebszustände der Schnittstellenkomponente *AC/DC Öffentliche Netz*

Betriebszustand	Beschreibung
Idle	Komponente ist betriebsbereit. Es findet kein Energietransfer statt.
Power from grid	Leistung wird vom öffentlichen Netz in das lokale Nano-Grid übertragen.
Power to grid	Leistung wird vom lokalen Nano-Grid in das öffentliche Netz übertragen.
Error	Komponente befindet sich im Sicherheitszustand.

Der Zustand *Power to grid* führt zu einem Energietransfer vom angeschlossenen öffentlichen Netz zum lokalen Nano-Grid. Der Zustand *Power to grid* führt zu einem Energietransfer vom lokalen Nano-Grid zum öffentlichen Netz.

3.3 Beschreibung der Steuer- und Regelstrategien

Die Steuer- und Regelstrategie für das Nano-Grid-Systemmodell ist hierarchisch und modular aufgebaut. In folgender Abbildung ist das Strategiekonzept mit den eingesetzten Hard- und Softwaremodulen dargestellt.

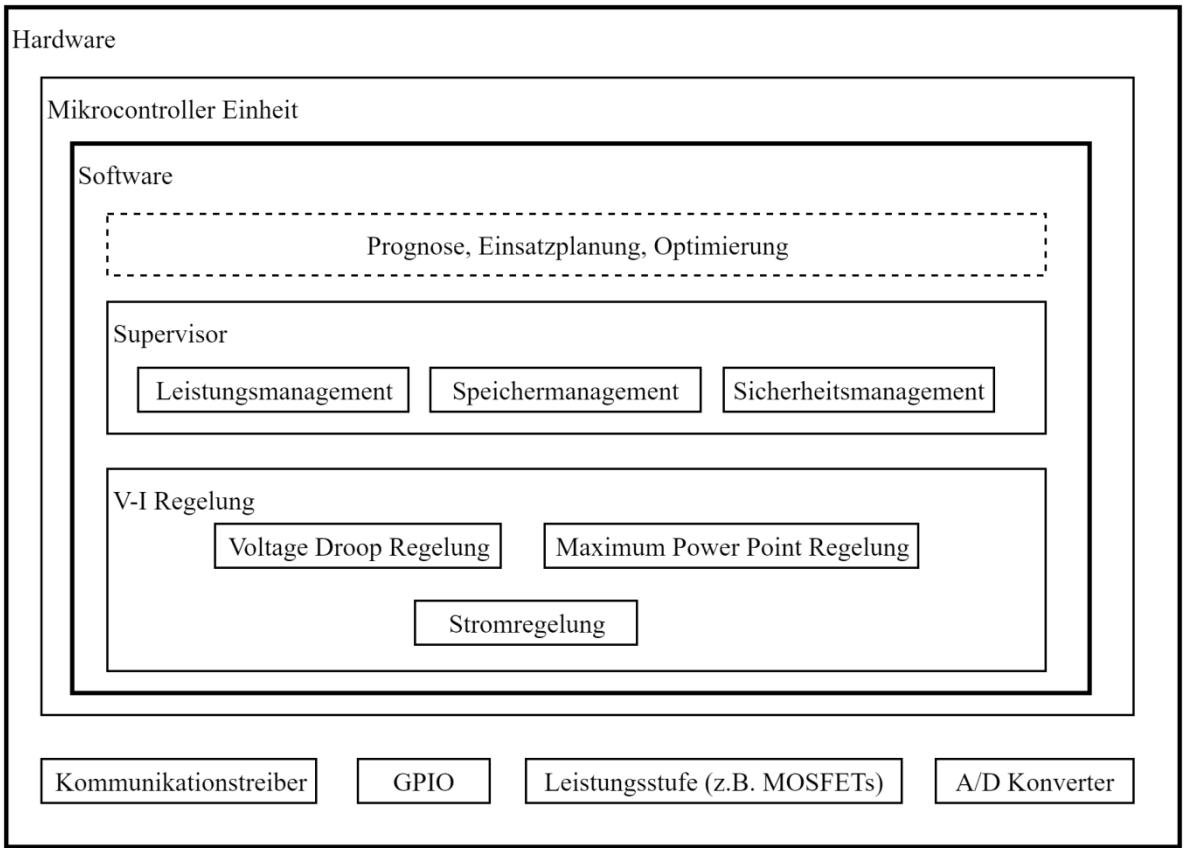

Abbildung 13: Modulübersicht des Hard- und Softwaremodells

Die Hardware ist in fünf Module unterteilt. Der Kommunikationstreiber ist für den Datenaustausch von Leistungswerten sowie Regel- und Steuersollwerten über einen Kommunikationsbus zuständig. Durch die programmierbaren Ein- und Ausgänge (englisch: General Purpose Input and Output, GPIO) wird der Lastschalter angesteuert sowie das globale Fehlersignal eingelesen. Die leistungselektronischen Stellglieder sind in der Leistungsstufe zusammengefasst. Für die Messwerterfassung von Strom und Spannung werden Analog-Digital Konverter eingesetzt. Die Mikrocontroller Einheit dient als Ablaufumgebung für die implementierte Software. Als Schnittstelle zur Leistungsstufe ist ein PWM-Ausgangssignal notwendig.

Die Softwaremodule setzen sich aus der Spannungs- und Stromregelung (V-I-Regelung), dem überlagerten Supervisor und einem übergeordneten Prognose-, Optimierung- und Einsatzplanungsmodul zusammen. Letzteres soll lediglich erwähnt werden, wird für den durchgeführten Steuerungsentwurf allerdings nicht weiter betrachtet.

Im Nachfolgenden wird die V-I-Regelung zum Verständnis der genutzten Steuerstrategien kurz erläutert werden. Anschließend werden die Spezifikationen in Form von Steuervorschriften informell beschrieben.

3.3.1 Spannungs- und Stromregelung

Die Funktion der V-I Regelung und des übergeordneten Leistungsmanagement ist die Ansteuerung der Leistungsstufe, welche durch folgendes Blockschaltbild ihrer Systemarchitektur beschrieben ist.

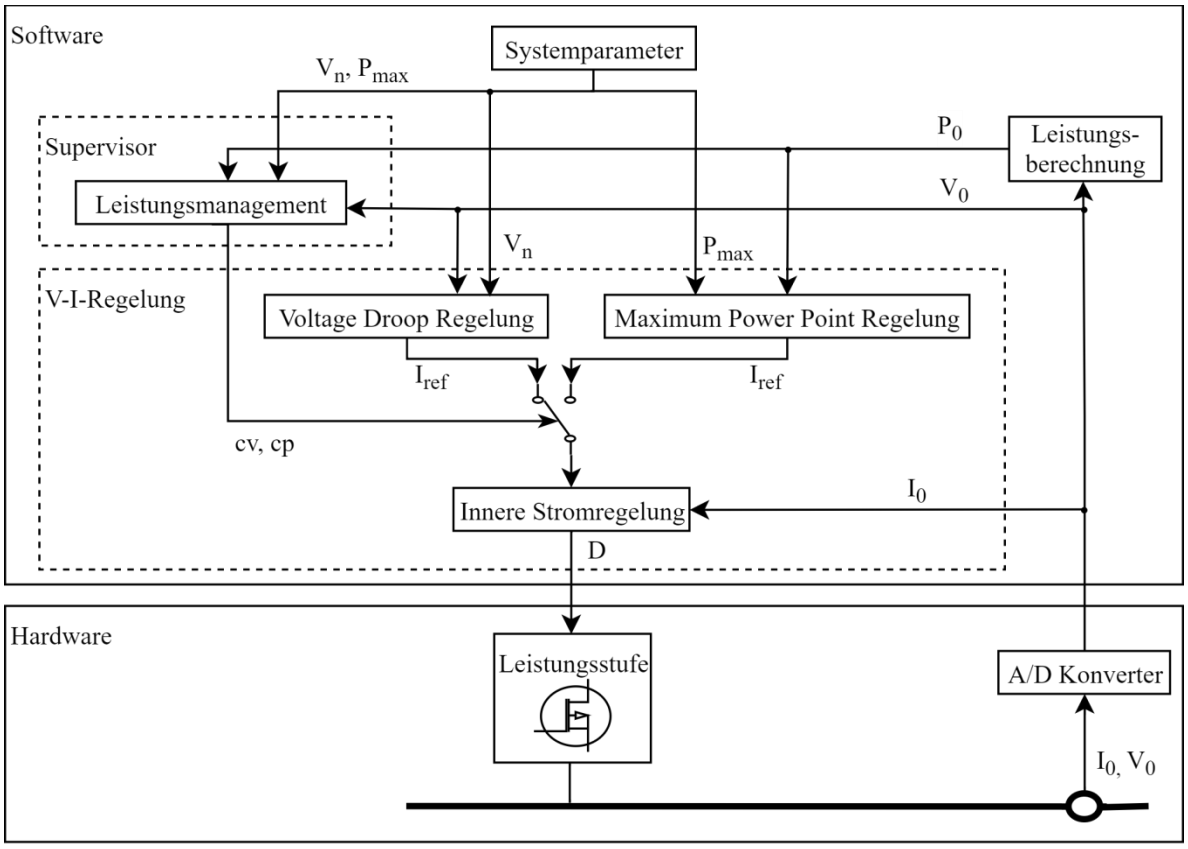

Abbildung 14: Blockschaltbild der hierarchischen Steuerungs- und Regelungsarchitektur

Der Spannungs- und Stromregler wird in Kaskade geschaltet. Der Spannungsregler ist dabei der äußere Regler, der als Führungsgröße eine Sollspannung erhält und als Regelgröße den Referenzstrom. Dieser wird an den nachgelagerten Stromregler als Sollwert übergeben,

welcher anschließend als Regelgröße den Schaltzyklus für die Ansteuerung der MOSFETs ausgibt. Beide Regler werden als PI-Regler realisiert.

Für die Spannungsvorgabe werden zwei Methoden verwendet. Die Voltage-Droop Methode (zu Deutsch: Spannungsabfall) und die Maximum Power Point Tracking (MPPT) Methode. Die Entscheidung, welche Methode und somit welcher Sollwert an den Spannungsregler übermittelt werden soll, wird vom übergeordneten Supervisor und dem darin enthaltenem Leistungsmanagement getroffen. Die MPPT Methode ist für jede Erzeugerkomponente speziell ausgelegt und wird daher nicht weiter betrachtet.

Die **Voltage-Droop Methode** gilt dagegen für alle DC/DC-Konverter gleich. Die Spannungsvorgabe erfolgt proportional zum Ausgangsstrom I_0 nach folgender Vorschrift:

$$V_0^* = V_n - k \cdot I_0. \tag{3.1}$$

Die Spannung V_n dient als Referenzwert für den jeweiligen Gleichspannungswandler und bestimmt, wann die Regelung aktiviert wird. Der Faktor k wird als virtueller Wiederstand genutzt und besitzt die Einheit $[\frac{V}{A}]$. Anhand von k wird der Grad der Lastverteilung zwischen den einzelnen Komponenten definiert. Je höher der Ausgangsstrom I_0, desto höher der Spannungsabfall $k \cdot I_0$. Der Sollwert V_0^* wird anschließend an den unterlagerten Spannungsregler übermittelt. In der nachfolgenden Abbildung ist ein Blockschaltbild dargestellt, welches die Voltage-Droop Methode für einen DC/DC-Konverter einer Erzeugerkomponente darstellt.

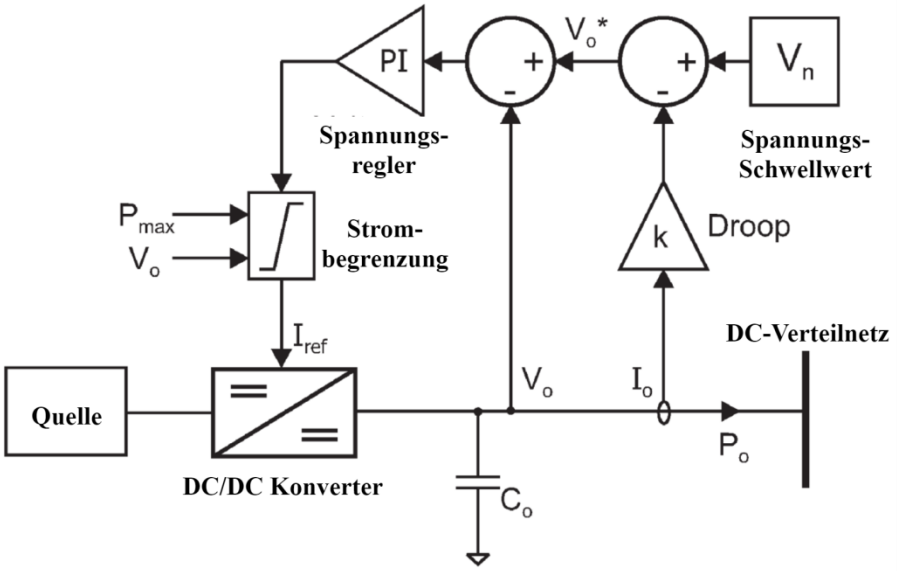

Abbildung 15: Blockschaltbild der Voltage-Droop-Regelung am Beispiel einer Erzeugerkomponenten (aus [33] übersetzt)

Der in Abbildung 15 dargestellte Spannungsregler wird als PI-Regler ausgeführt und ist in Feed-Forward Architektur aufgebaut. Die Regeldifferenz aus der Sollwertgröße V_0^* und dem aktuellen Spannungsistwert V_o ist die Eingangsgrößen des Reglers. Folgende Vorschrift beschreibt den Spannungsregler im Laplace-Bereich:

$$I_{ref} = (V_0^* - V_o)(k_{pV} + k_{iV}\frac{1}{s}). \tag{3.2}$$

Der nachgestellte Stromregler wird durch die maximale Leistungsvorgabe der einzelnen Betriebszustände dynamisch nach

$$I_{ref,max} = \frac{P_{max}}{V_0} \tag{3.3}$$

begrenzt. Die maximale Leistung P_{max} wird durch die Betriebszustände für die jeweiligen Gleichspannungswandler definiert. Der Stromsollwert I_{ref} wird nach Formel (3.2) gebildet und anschließend an den PI-Stromregler weitergegeben. Dieser besitzt folgende Reglervorschrift:

$$D = \left(I_{ref} - I_o\right)(k_p + k_i\frac{1}{s}). \tag{3.4}$$

Die Ausgangsgröße des Stromreglers ist der Schaltzyklus D. Dieser beschreibt die Schaltfrequenz, mit welcher die Leistungsschalter respektive der MOSFETs in den DC/DC-Konvertern angesteuert werden. Der Verstärkungsfaktor wird durch k_p beschrieben und das Integralglied durch k_i, welches die Nachstellzeit T_i der PWM-Strecke mit $1/T_i$ kompensiert.

Der Buskondensator C_0 stellt einen integrierten Pufferkondensator in den DC/DC-Konvertern dar. Alternativ kann auch eine Batterie als Pufferkondensator verwendet werden. Diese Variante ist anlog zu dem direkten Anschluss einer Batterie an das Verteilnetz und wird in dieser Arbeit nicht berücksichtigt. Wichtig bei der Auslegung der Pufferstrecke ist, dass die Transienten bei einem plötzlichen Schalten von Erzeugern oder Lasten so gering wie möglich gehalten werden. Dies kann durch elektronische Bauteile und durch eine dynamische Regelung der PWM Strecke erfolgen.

Der V-I-Regleraufbau ist für alle DC/DC-Konverter identisch. Der Strom I_o bei den Erzeugerkomponenten positiv, bei den Lastkomponenten negativ. Die Speicherkomponenten sowie die Schnittstellenkomponenten sind bidirektional ausgeführt. Damit ist der Stromfluss sowohl negativ als auch positiv möglich.

3.3.2 Leistungsmanagement

Das Leistungsmanagement wird in das Erzeuger-, Last- und Schnittstellenmanagement unterteilt.

Das **Erzeugermanagement** steuert explizit die Erzeugerkomponenten nach einer vorgegebenen Ablaufplanung an. Dafür werden die Komponenten *DC/DC Regenerativ* und *DC/DC Nicht-Regenerativ* nach einer definierten Priorität geschalten. Die Speicherkomponente *DC/DC Speicher* wird ebenfalls in den Ablaufplan mit einbezogen, da diese sowohl einen Erzeuger als auch einen Verbraucher darstellt. Die Priorität der Zuschaltung der Erzeugerkomponenten ist folgendermaßen definiert.

- Prio 1: Zuschalten von *DC/DC Regenerativ*
- Prio 2: Entladen durch *DC/DC Speicher*
- Prio 3: Zuschalten von *DC/DC Nicht-Regenerativ*

Die nicht erneuerbaren Energiequellen sollen erst eingeschaltet werden, wenn weder die erneuerbare Quelle noch der Speicher die benötigte Lastleistung bereitstellen können. In dieser Auslegung ist eine feste Priorität hinterlegt. Es ist auch möglich eine dynamische Priorität zu hinterlegen. Dabei können Gewichtungsfaktoren in die Entscheidung integriert werden. Die Gewichtungsfaktoren werden in der übergeordneten Prognoseberechnung bestimmt (siehe Abbildung 13).

Folgende Tabelle gibt eine Übersicht über die Ablaufreihenfolge. Je nach Leistungszustand werden die verschiedenen Betriebsarten aus Kapitel 3.2 der einzelnen Konverter geschaltet. Diese geben den entsprechenden Sollwert an die unterlagerte Regelung weiter.

Tabelle 7: Steuervorschrift für das Erzeugermanagement

Schwellwerte	**Zustand**	**DC/DC Regenerativ**	**DC/DC Nicht- Regenerativ**	**DC/DC Speicher**
$P_L < P_R$	1	Constant Voltage	Off	Charge
$P_L > P_R$	2	Constant Power	Off	Discharge
$P_L > P_R + P_S$	3	Constant Power	Constant Voltage	Discharge
$P_L > P_R + P_S + P_N$	4	Constant Power	Constant Power	Discharge

Entscheidend für die Priorisierung ist der Vergleich zwischen der Lastleistung P_L und der maximal verfügbaren Leistung der eingeschalteten Erzeugerkomponenten. Die Lastleistung P_L beschreibt die Summe aus der mit einem Schalter angeschlossene Lastleistung P_{L2} und der mit dem Konverter gekoppelten Lastleistung P_{L1}. Im Zustand 1 ist mehr regenerative Leistung

P_R verfügbar als benötigt wird. Die Erzeugerkomponente *DC/DC Regenerativ* wird auf die Betriebsart *Constant Voltage (CV)* geschaltet. Damit wird die Voltage-Droop Regelung aktiviert. Die nicht erneuerbare Quelle *DC/DC Nicht- Regenerativ* ist ausgeschaltet und der angeschlossene Speicher wird außerdem über die Betriebsart *Charge* geladen. Sollte der Speicher vollgeladen sein, wird dies durch die separate Spezifikation Storage Management (siehe Kapitel 3.3.3) verhindert.

Sobald die Lastleistung die verfügbare regenerative Leistung überschreitet, wechselt der Systemzustand auf Zustand 2. In diesem Zustand schaltet *DC/DC Regenerativ* auf *Constant Power (CP)*, welches die Regelung auf die MPPT Methode setzt. Die Komponente *DC/DC Nicht- Regenerativ* bleibt weiterhin ausgeschaltet. Die Differenz zwischen erneuerbarer Leistung und Lastleistung wird durch das Entladen des Speichers bereitgestellt. Steigt die Lastleistung weiter an und überschreitet die Summe aus regenerativer Leistung P_R und der Speicherleistung P_S, wird im Zustand 3 die nicht regenerative Quelle *DC/DC Nicht-Regenerativ* im *Constant Voltage* Modus dazu geschaltet. Die Betriebsart für *DC/DC Regenerativ* und *DC/DC Speicher* bleibt im vorherigen Betriebszustand. Sobald die Lastleistung die Summe der maximalen Leistung aller verfügbaren Erzeugerkomponenten übersteigt, wird im Zustand 4 die Komponente *DC/DC Nicht- Regenerativ* auf *Constant Power* geschaltet. Ohne Abschaltung von Lasten oder Zuschalten von externen Quellen würde die Netzspannung einbrechen. Um das zu verhindern, werden ein Lastmanagement sowie ein Schnittstellenmanagement implementiert

Das **Lastmanagement** ist für die priorisierte Abschaltung der angeschlossenen Lasten zuständig. Für die Abschaltung sind zwei unterschiedliche Stellglieder verfügbar. Über den Gleichspannungswandler *DC/DC Last* kann der abgegebene Strom auf einen vorgegebenen Wert begrenzt werden oder ganz ausgeschaltet werden. Durch den Lastschalter kann die Last lediglich ein- oder ausgeschaltet werden. Die Priorität der Abschaltung ist wie folgt gegeben. Es ist zu beachten, dass die Priorität eine umgekehrte Reihenfolge zum Erzeugermanagement aufweist.

- Prio 1: Abschalten von *DC/DC Last*
- Prio 2: Ausschalten des Lastschalters

Die über den Lastschalter angeschlossene Last hat in diesem Fall eine höhere Priorität als die über einen Koverter gekoppelte Last. Damit wird die direkt angeschlossene Last erst abgeschaltet, wenn der Leistungsbedarf trotz Abschaltung von *DC/DC Last* nicht gedeckt ist. Folgende Tabelle stellt den Ablaufplan für das Lastmanagement dar. Die Nummerierung der Zustände wird aufbauend zur Nummerierung vom Erzeugermanagement fortgesetzt.

Tabelle 8: Steuervorschrift für das Lastmanagement

Leistungsvergleich	Zustand	DC/DC Last	Lastschalter
$P_L > P_R + P_S + P_N$	4	Limit	Equal
$P_L > P_R + P_S + P_N$	5	Off	Equal
$P_L > P_R + P_S + P_N$	6	Off	Off

Die Lastleistung wird weiterhin mit der Summenleistung P_L beschrieben. Anknüpfend an die zuvor priorisierte Zuschaltung der nicht erneuerbaren Erzeugerkomponente im Zustand 4, wird bei Überschreiten der verfügbaren Gesamterzeugerleistung $P_R + P_S + P_N$ der abgegebene Strom der Komponente *DC/DC Last* begrenzt. Dies erfolgt durch Begrenzung der maximalen Leistung P_{max}, welche nach Gleichung (3.3) im Stromregler den maximalen Strom I_{max} begrenzt. Der Lastschalter kann ein- oder ausgeschaltet sein, was mit dem Zustand *Equal* beschrieben ist. Sollte trotz Strombegrenzung die Lastleistung weiterhin nicht gedeckt sein, wird *DC/DC Last* in Zustand 5 ausgeschalten. Der Lastschalter kann weiterhin ein- oder ausgeschaltet bleiben. In Zustand 6 wird nun auch der Lastschalter abgeschaltet, sofern die Lastleistung, welche in diesem Zustand nur noch aus der Leistung der direkt angeschlossenen Last besteht, nicht gedeckt werden kann. Durch diese Ablaufreihenfolge der Lastabschaltung wird ein Einbruch der Netzspannung verhindert.

Das **Schnittstellenmanagement** steuert die Schnittstellenkomponenten *DC/DC Schnittstelle* und *AC/DC Öffentliches Netz* nach einer vorgegebenen Steuervorschrift an. Anders als beim Erzeuger- und Lastmanagement wird nicht die Lastleistung als Vergleichsgröße herangezogen, sondern es wird ein Gesamtzustand der verschalteten Nano-Grids definiert. Der Gesamtzustand eines Nano-Grids wird anhand der drei Zustandsadjektive *saturated* (zu Deutsch: gesättigt), *deficient* (zu Deutsch: mangelhaft) und *self-sufficient* (zu Deutsch: selbständig) beschrieben.

Tabelle 9: Beschreibung der Nano-Grid Gesamtzustände für das Schnittstellenmanagement

Zustand	Beschreibung
saturated	Nano-Grid besitzt mehr Leistung und Energie als es für die eigene Lastversorgung benötigt
Self-sufficient	Nano-Grid besitzt genügend Leistung und Energie um sich selbst zu versorgen
deficient	Nano-Grid besitzt nicht genügend Leistung und Energie um die Lasten zu versorgen

Die Kriterien wann ein Nano-Grid in einen dieser Zustände schaltet, kann vom Anwender selbst festgelegt werden. In dieser Modellbetrachtung ist das Nano-Grid *self-sufficient*, solange die Energiequellen (auch die nicht erneuerbaren Quellen) die Lastanforderungen

abdecken. Ist dies nicht der Fall, so ist das Nano-Grid *deficient*. Sobald die erneuerbaren Quellen mehr Leistung liefern als die angeschlossenen Lasten benötigen und auch die Speicherkomponente vollgeladen ist, ist das Nano-Grid *saturated*.

Anhand dieser Zustände werden das lokale Nano-Grid und das angeschlossene externe Nano-Grid bzw. das Verbundnetz (nachfolgend als externes System zusammengefasst) gegenübergestellt. Für das Verbundnetz gelten die Zustände aus Tabelle 9 entsprechend. Folgende Steuervorschrift wird in Form einer Entscheidungsmatrix entworfen:

Tabelle 10: Steuervorschrift für das Schnittstellenmanagement

lokales Nano-Grid	**Externes System**	**DC/DC Schnittstelle**	**AC/DC Öffentliches Netz**
saturated	saturated	Idle	Power to Grid
saturated	self-sufficient	Power external	Idle
saturated	deficient	Power external	Idle
self-sufficient	saturated	Power internal	Idle
self-sufficient	self-sufficient	Idle	Idle
self-sufficient	deficient	Power external	Idle
deficient	saturated	Power internal	Idle
deficient	self-sufficient	Power internal	Idle
deficient	deficient	Idle	Power from Grid

Sind beide System im Zustand *saturated*, ist die Komponente *AC/DC Öffentliches Netz* im Betriebszustand *Power to Grid* und die überschüssige Leistung wird in das öffentliche Netz gespeist. Alternativ zum öffentlichen Netz kann auch eine Wärmespumpe oder eine Power-to-Heat Anlage mit dem überschüssigen Strom versorgt werden, um gegebenenfalls ein Wärmenetz zu versorgen. Die Komponente *DC/DC Schnittstelle* läuft im Betriebszustand *Idle*. Ist das lokale Nano-Grid gesättigt und das externe System nicht gesättigt, schaltet *DC/DC Schnittstelle* auf *Power to external* und versorgt das externe System mit Strom aus dem lokalen Nano-Grid zu versorgen. Das Öffentliche Netz wird nicht in Anspruch genommen. Kann sich das lokale Nano-Grid selbstständig versorgen, wird abhängig vom externen Systemzustand Ladung nach intern oder nach extern ausgetauscht. Sind beide Nano-Grid-Systeme im Zustand *self-sufficient* erfolgt kein Energietransfer. Sobald beide Nano-Grid-Systeme im Zustand *deficient* sind, wird Strom aus dem öffentlichen Netz bezogen.

3.3.3 Speichermanagement

Neben dem Leistungsmanagement, welches für die Ansteuerung der einzelnen DC/DC-Konverter verwendet wird, besitzt das Nano-Grid auch ein Speichermanagement. Dieses schützt die Batterie vor Überladung und Tiefenentladung. Der relevante Parameter für die Steuerung ist der SOC des angeschlossenen Speichermoduls. Dieser wird abhängig von der Speichertechnologie unterschiedlich ermittelt. Sobald ein definierter Grenzwert über- oder unterschritten wird, verhindert die Steuerung das Entladen oder Laden der Speicherkomponente.

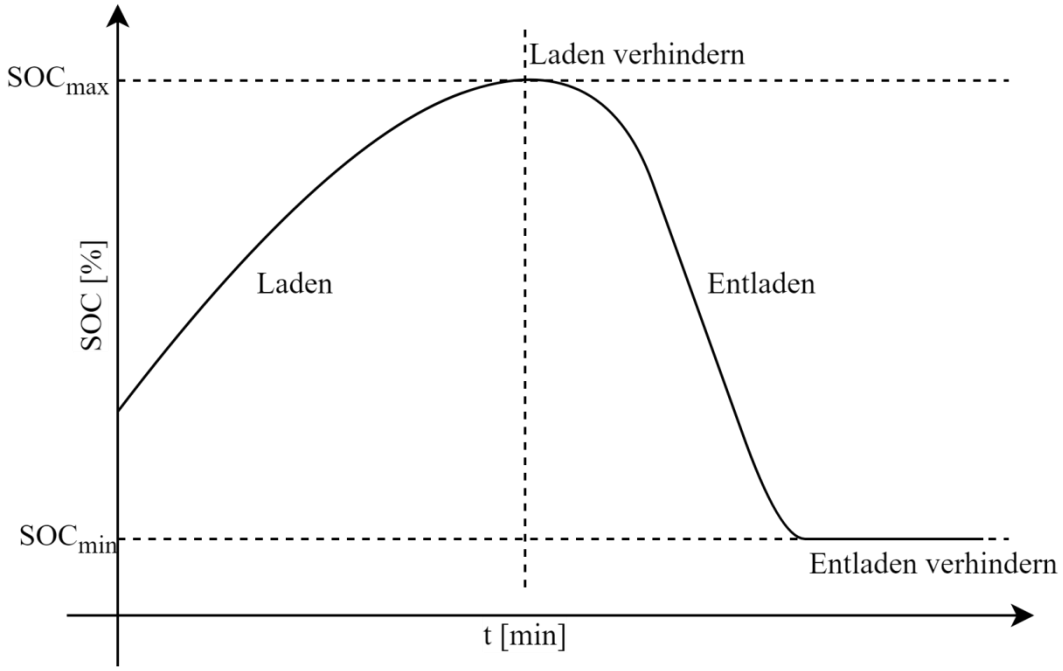

Abbildung 16: Schwellwerte des State of Charge (SOC)

Die Grenzwerte werden in der Steuerung mit SOC_{max}und SOC_{min} definiert. Das Speichermanagement ist auf der Komponente *DC/DC Speicher* implementiert.

3.3.4 Sicherheitsmanagement

Das Sicherheitsmanagement steuert den Anlaufprozess, nachdem das globale Fehlereignis aufgetreten ist und sich alle Komponenten im Fehlerzustand befinden. Die einzelnen Komponenten werden bezüglich ihrer Priorisierung nacheinander vom Zustand *error* in die Zustände *Idle* oder *Off* überführt. Das folgende Ablaufdiagramm stellt die festgelegte Reihenfolge dar.

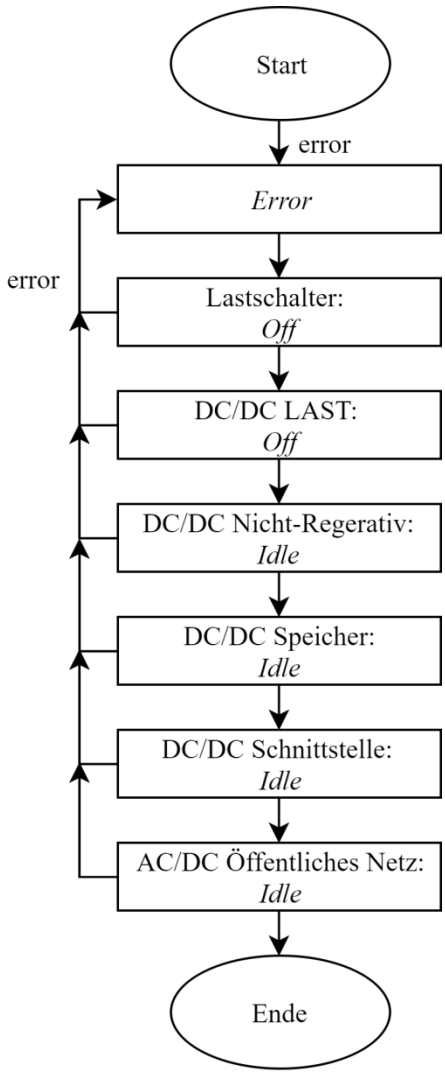

Abbildung 17: Ablaufplanung des Sicherheitsmanagements

Alle Komponenten befinden sich durch Eintreten des globalen Fehlersignals *error* im Zustand *Error*. Um nach Quittierung des Fehlers das System wieder in die Initialzustände zu führen, werden im ersten Schritt die Lasten beginnend mit dem Lastschalter abgeschaltet. Anschließend werden die Erzeugerkomponenten betriebsbereit gesetzt. Danach wird der Speicher und abschließend die Schnittstellenkonverter in den Zustand *Idle* gesetzt. Sobald während dem Ablauf erneut das Ereignis *error* auftritt, springt die Ablaufkette zurück zum

Zustand *Error*. Im Zustand *Error* werden bereits alle sicherheitsrelevanten Parameter der einzelnen Komponenten gesetzt. Diese sind frei definierbar. In der Realität sollten die Maßnahme aktiviert werden, welche die angeschlossenen Geräte schützen.

3.1 Beschreibung der Steuerungsarchitekturen

Für die Steuerung und Regelung des lokalen Nano-Grid-Systems werden drei verschiedene Steuerungsarchitekturen gegenübergestellt. Diese unterscheiden sich ihrer Kommunikationstopologie, welche entscheidend für den Informationsaustausch der Zustände zwischen den einzelnen DC/DC-Konvertern ist. Es wird eine zentrale sowie verteilte Architektur vorgestellt.

3.1.1 Zentrale Steuerungsarchitektur

In der zentralisierten Topologie ist eine zentrale Steuerung für das Nano-Grid Management zuständig. Hierfür eignet sich beispielsweise ein Einplatinenrechner mit entsprechendem Kommunikationstreiber. Folgende Abbildung stellt beispielhaft für drei DC/DC-Konverter den Systemaufbau dar.

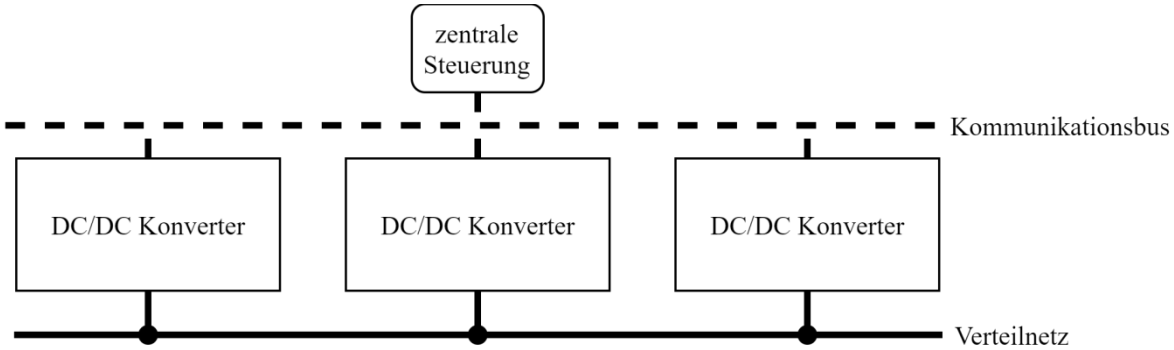

Abbildung 18: Systemübersicht der zentralen Steuerungsarchitektur

In diesem Aufbau besitzt jeder Gleichspannungswandler (hier vereinfacht als DC/DC-Konverter dargestellt) einen Kommunikationstreiber, welcher mit einer zentralen Steuereinheit kommuniziert. In der abstrahierten Modellbetrachtung ist die eingesetzte Bustechnologie irrelevant. Als Beispiel soll ein Controller Area Network (CAN) als serieller Kommunikationsbus genannt werden. Es können Daten sowohl gelesen als auch geschrieben werden. Zu den gelesenen Daten gehören der auf den Konvertern gemessene Strom und Spannung. Zu den schreibenden Parametern gehören die Sollwertvorgaben für die jeweiligen

Spannungsregler in den DC/DC-Konvertern. Der elektrische Ladungsaustausch erfolgt über das lokale Verteilnetz.

Vorteil dieser Strategie ist die einfache Implementierung, da nur eine Ablaufumgebung notwendig ist. Nachteil ist die Abhängigkeit von einer einzigen Steuereinheit. Sollte diese ausfallen, ist das gesamte Nano-Grid nicht ablauffähig. Die zentrale Steuerung dient als Grundlage für den monolithischen Steuerungsentwurf.

3.1.2 Verteilte Steuerungsarchitektur

Eine weitere Strategie ist die verteilte Steuerung der einzelnen Komponenten. Hier wird davon ausgegangen, dass jeder DC/DC-Konverter eine Steuereinheit in Form eines Mikrocontrollers besitzt, auf dem die lokalen Regel- und Steueralgorithmen ablaufen (siehe Abbildung 19).

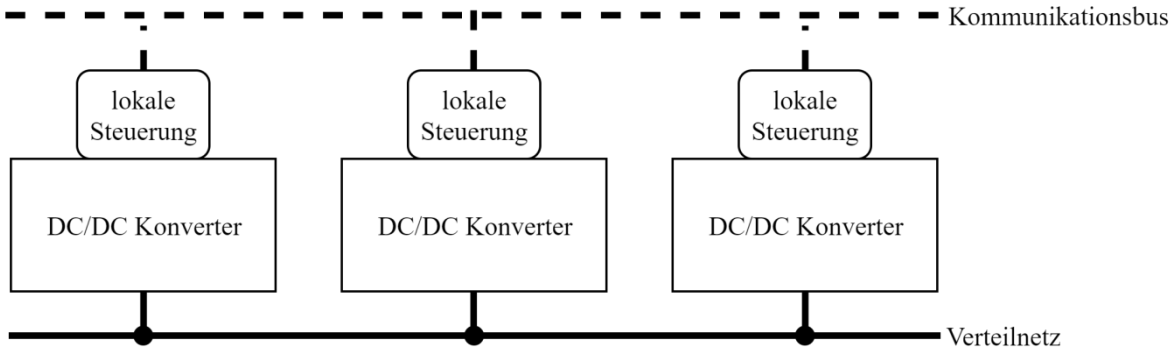

Abbildung 19: Systemübersicht der verteilten Steuerungsarchitektur

Über den Kommunikationsbus werden die jeweiligen lokalen Istwerte an die weiteren angeschlossenen Steuereinheiten übertragen. Der Leistungsaustausch zwischen den Komponenten erfolgt über Verteilnetz.

Der Vorteil gegenüber der zentralen Variante ist, dass bei Ausfall einer Steuereinheit die anderen DC/DC-Konverter weiterhin funktionsfähig sind. So kann beispielsweise bei Ausfall einer Erzeugerkomponente, weiterhin Ladung durch den angeschlossenen Speicher bereitgestellt werden. Damit ist eine höhere Verfügbarkeit gegeben. Nachteil ist der leicht erhöhte Implementierungsaufwand. Außerdem ist eine Abhängigkeit bezüglich des Kommunikationsbusses gegeben.

Als besondere Form der verteilten Steuerungsarchitektur wird das DC-Bus Signaling (DBS) eingesetzt. Dieses wird in [32] ausführlich beschrieben. Es bildet eine Erweiterung zur oben beschriebenen verteilten Architektur. Folgende Abbildung zeigt den angepassten Systemaufbau.

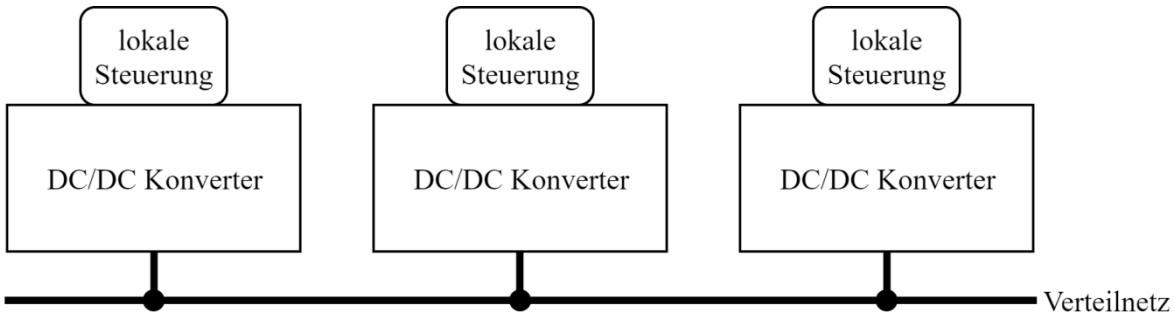

Abbildung 20: Systemübersicht der DC-Bus Signaling Architektur

Jeder Konverter besitzt eine lokale Steuereinheit zum Ablauf der Steuer- und Regelalgorithmen. Der Informationsaustausch erfolgt allerdings nicht über einen Kommunikationsbus sondern über das Verteilnetz. Die Spannung V_{Grid} wird genutzt um den Zustand des Nano-Grids hinsichtlich des Lastleistungsbedarfs zu beschreiben. Durch Definition definierter Spannungsgrenzwerte wird der Zustand bezüglich des erzeugten und benötigten Stromes beschrieben.

Für das DBS ist eine interne Voltage-Droop Regelung in den einzelnen DC/DC-Konvertern notwendig, die den Spannungsausgang abhängig vom Ausgangstrom nach Gleichung (3.1) regeln (siehe Kapitel 3.3.1). In folgender Abbildung wird der Zusammenhang zwischen dem Spannungswert und den Zuständen 1 und 2 aus Tabelle 7 deutlich.

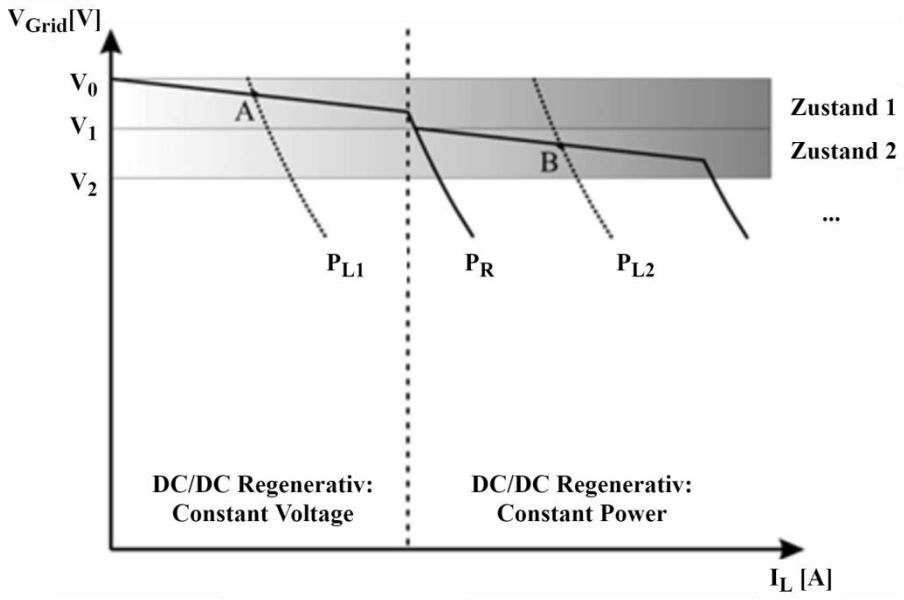

Abbildung 21: Betriebszustände der regenerativen Erzeugerkomponente abhängig der Verteilnetzspannung unter Einsatz von DC-Bus Signaling [33]

Im Arbeitspunkt A ist die Lastleistung P_{L1} kleiner als die regenerative Erzeugerleistung P_R und die Spannung V_{Grid} liegt über dem Schwellwert V_1. Das System befindet sich im Zustand

1. Im Arbeitspunkt B wird die zweite Last dazu geschaltet. Die Erzeugerleistung P_R ist nun kleiner als die Lastleistung und die Spannung V_{Grid} liegt unter dem Schwellwert V_2. Somit befindet sich die Komponente *DC/DC Regenerativ* im Betriebszustand *Constant Power*.

Jede Über- oder Unterschreitung einer Spannungsgrenze überführt das System sequentiell in den nächsten Zustand. Die Grenzwerte sind für jeden Konverter zu definieren. Für die Berechnung der Grenzwerte V_n müssen nach [35] folgende Parameter nach der Vorschrift

$$V_n = V_{n-1} - V_{d_n} - V_e \tag{3.5}$$

berücksichtigt werden. Für die Grenzwertberechnung wird vom vorherigen Grenzwert V_{n-1} eine Toleranzspannung V_{d_n}, welche den maximalen Spannungsabfall der angewendeten Voltage-Droop Regelung darstellt und V_e, welche den Messfehler und die maximal möglichen Transienten darstellt, subtrahiert.

Die Ablaufplanungen für das Erzeuger- und Lastmanagement aus Kapitel 3.3.2 können damit in Abhängigkeit der Verteilnetzspannung V_{Grid} gesteuert werden. Die Netzspannung kann von jeder Systemkomponente gemessen werden. Somit kann auf einen zentralen Kommunikationsbus verzichtet werden. Die DC/DC-Konverter benötigen keinen Kommunikationstreiber, was zu einer kostengünstigeren Systemauslegung führt. Zusätzlich kann das System sehr einfach modular erweitert werden. Nachteil ist die begrenzte Anzahl an Modulen, da nur ein gewisser Spannungsbereich für die Grenzwerte und die jeweiligen Toleranzen zur Verfügung steht. Für kleinskalierte Nano-Grids ist diese Steuerungsarchitektur sehr gut geeignet. Das DBS Verfahren dient als Grundlage für den lokal-modularen Steuerungsentwurf.

4.Konzeption

In diesem Kapitel wird die Auswahl der genutzten Beschreibungsform, des durchgeführten Steuerungsentwurfes sowie der Entwicklungsumgebung begründet. Die verschiedenen Kriterien werden tabellarisch gegenübergestellt. Jedes Kriterium wird ein Wert von 1 (mit -- markiert) bis 5 (mit ++ markiert) zugewiesen, welcher zusätzlich mit einem Relevanzfaktor gewichtet wird. Das Gesamtergebnis ist in der jeweiligen letzten Zeile dargestellt.

4.1 Auswahl der Entwicklungsumgebung

Die Entwicklungsumgebung wird für die Modellbildung der ereignisdiskreten Strecke und Spezifikation sowie für den Steuerungsentwurf verwendet. Daher sollten die Entwurfsalgorithmen nach der SCT korrekt implementiert sein. Zur Auswahl stehen das DESTool, welches am Lehrstuhl für Regelungstechnik an der Friedrich-Alexander Universität Erlangen-Nürnberg entwickelt wird [30], das TCT Tool der Universität Toronto [17] sowie die SPN Tool Box für Matlab, welches an der Universität Notre-Dame in Paris entwickelt wird [36]. Eine Gegenüberstellung der Softwaretools ist in folgender Tabelle zusammengefasst.

Tabelle 11: Bewertungsmatrix für die Auswahl der Entwicklungsumgebung

Kriterium	**Relevanz**	**DESTool**	**TCT**	**SPN Toolbox**
Grafische Darstellbarkeit	5	5 (++)	3 (o)	1 (--)
Bedienbarkeit	4	5 (++)	1 (--)	2 (-)
Integrierte Funktionalität	5	5 (++)	5 (++)	5 (++)
Simulationsmöglichkeiten	4	4 (++)	3 (o)	3 (o)
Open Source Lizenzierung	5	5 (++)	3 (o)	1 (--)
Codegenerator verfügbar	5	3 (++)	5 (++)	5 (++)
	Ergebnis	**126**	**96**	**80**

Die grafische Darstellbarkeit bietet dem Anwender eine gute Übersicht bei der visuellen Analyse der erstellten Modelle und ermöglicht nachfolgenden Nutzern einen entscheidenden Vorteil hinsichtlich der Nachvollziehbarkeit des Systems. Die grafische Darstellung beim TCT Tool ist durch das eingeschränkte Benennen der Zustände und Ereignisse unbequem. Bei der SPN Tool Box ist die Darstellung nur in Matrizenschreibweise verfügbar. Das DESTool bietet hier die besten Darstellungsmöglichkeiten durch die integrierte Graphviz Bibliothek, welche eine saubere und klare Visualisierung der erstellten Generatoren inklusive der individuellen Nennung der Zustände und Ereignisse zulässt.

Die Bewertung der Bedienbarkeit der Software wird anhand der Eingabemöglichkeiten der notwendigen Parameter für die Erstellung eines Generators gemessen. Dazu gehören auch die Korrekturmöglichkeiten von Eingabefehler. Hier bietet das DESTool den komfortabelsten Umgang.

Die integrierten Funktionen beziehen sich auf die hinterlegten Analyse- und Entwurfsalgorithmen. Hier haben alle drei Entwicklungsumgebungen einen gleichwertigen Funktionsumfang.

Durch die integrierten Simulationsmöglichkeiten können die entworfenen Steuerungen verifiziert werden. Das DESTool bietet durch seine separate Benutzeroberfläche für die Simulation die beste Handhabbarkeit.

Die Bereitstellung der Software durch eine Open Source Lizenz ist eine allgemeine Anforderung dieser Arbeit. Dadurch soll eine offene Kollaborationsmöglichkeit gegeben werden. Auch in diesem Kriterium hat das DESTool durch die nicht restriktive Open Source Lizenz *LGPL* einen Vorteil gegenüber den anderen Softwaretools. Das TCT Tool ist zwar kostenfrei verfügbar, jedoch sind die implementierten Bibliotheken nicht quelloffen. Die kostenlose SPN Toolbox wird durch die notwendige Matlab Umgebung in ihrer freien Verfügbarkeit eingeschränkt.

Das abschließende Kriterium der verfügbaren Codegeneratoren wird für den Implementierungsschritt des modellbasierten Steuerungsentwurfes in eine reale Ablaufumgebung benötigt. Die Anforderung für diese Arbeit ist es Programmcode für eingebettete Systeme, in diesem Fall die Steuereinheiten für Gleichspannungswandler, zu erhalten. Für das DESTool wurde im Jahr 2014 im Rahmen einer Masterarbeit ein Codegenerator entwickelt, welcher aus den entworfenen Generatoren einen IEC61131 konformen Programmcode erzeugt. Das TCT Tool gibt seinen Generatorcode direkt in der Programmiersprache C aus. Der Codegenerator der SPN Tool Box ist durch die Integration in Matlab gekennzeichnet. Die Wahl der Entwicklungsumgebung ist nach der gesamten Bewertung auf das DESTool gefallen.

4.2 Auswahl der Beschreibungsform

Für die Wahl der Beschreibungsform stehen wie in Kapitel 2.2.3 erläutert, stehen DFA und Petrinetze zur Auswahl. Folgende Kriterien werden tabellarisch gegenübergestellt.

Tabelle 12: Bewertungsmatrix für die Auswahl der Beschreibungsform

Kriterium	**Relevanz**	**Automaten**	**Petrinetze**
Sequentielle Prozessdarstellung	4	5 (++)	5 (++)
Parallele Prozessdarstellung	3	3 (0)	5 (++)
Entwicklungsumgebung vorhanden	5	5 (++)	5 (++)
Nutzung von DESTool	5	5 (++)	1 (--)
	Ergebnis	**79**	**65**

Beide Beschreibungsformen sind in gleich mächtig ereignisdiskrete Prozesse darzustellen. Mit Petrinetzen können parallele Prozesse in einem Generatormodell abgebildet werden. Bei DFA kann jeder Prozess nur einzeln in einem Generator beschrieben werden. Durch parallele Komposition können diese aber miteinander gekoppelt werden. In dieser Arbeit werden die einzelnen Systemkomponenten durch eine sehr feinmodulare Dekomposition dargestellt, daher ist die parallele Darstellung der Prozess nicht sehr hoch gewichtet.

Aufgrund der getroffenen Auswahl der Entwicklungsumgebung ist es notwendig, dass die gewählte Beschreibungsform im DESTool modelliert werden kann. DESTool bietet ausschließlich die Modellierung über Automaten an, daher fällt die Wahl der Beschreibungsform auf die Automaten.

4.3 Auswahl des Entwurfsansatzes

Für den Steuerungsentwurf nach SCT stehen der monolithische, lokale, lokal-modulare und dezentrale Entwurfsansatz zur Auswahl. Die für diese Arbeit relevanten Unterscheidungsmerkmale der verschiedenen Entwurfsansätze sind die algorithmische Komplexität, welche aus der Modellkomplexität abgeleitet wird, die globale Bereitstellung von Signalen sowie die Möglichkeit der verteilten Implementierung.

Für das Sicherheitsmanagement wird der monolithische Entwurfsansatz gewählt, da durch das Fehlerereignis *error* global verfügbar sein muss. Es wird eine zentrale Sicherheitssteuerung

unter Nutzung der zentralen Steuerungsarchitektur entworfen. Alternativ kann auch der lokale Entwurfsansatz unter Nutzung der verteilten Steuerungsarchitektur verwendet werden. Hierfür müssen die Signale über einen Kommunikationsbus global zur Verfügung stehen.

Für das Leistungsmanagement wird unter Nutzung der DBS Methode der lokal-modulare Steuerungsentwurf durchgeführt. Der Betriebszustand *Error* wird dabei nicht berücksichtigt. Dies lässt eine verteilte Implementierung zu bei der jede Systemkomponente lokal ihre Steuerungsaufgabe ausführt. Dadurch kann das Streckenmodell sowie die Spezifikationen sehr strukturiert abgebildet werden. Für die Komponente *DC/DC Speicher* wird das Speichermanagement mit dem Erzeugermanagement zusammengeführt.

Der dezentrale Entwurfsansatz wird nicht angewendet, da alle Steuersignale im System beobachtbar sind.

5.Modellierung und Steuerungsentwurf

In diesem Kapitel werden das ungesteuerte Systemverhalten des betrachteten Nano-Grid-Modells sowie die Steuerungsaufgaben in Form von Spezifikationen formal als Generatoren beschrieben. Die Abstrahierung führt zu einem logischen ereignisdiskreten Verhaltensmodell. Abschließend wird der resultierende Steuerungsentwurf beschrieben. Es werden allgemein folgende Modellannahmen für die Modellierung getroffen:

- Es wird nur das logische Verhalten berücksichtigt
- Es wird davon ausgegangen, dass die unterlagerte Regelung von Strom und Spannung korrekt funktioniert.
- Es sind nur die genannten Komponenten am Verteilnetz angeschlossen.
- Zu Beginn befinden sich sämtliche Komponenten in ihrem definierten Initialzustand.

5.1 Modell der ungesteuerten Strecken

Die Modellkomponenten der ungesteuerten Strecke werden generell in zwei Typen unterteilt. Ein Typ beschreibt die einzelnen DC/DC-Konverter und wird als aktive Steuerkomponente betrachtet. Sie enthalten steuerbare Ereignisse, welche durch die Spezifikation beschränkt oder in der Ablaufreihenfolge vorgegeben werden können. Der zweite Typ an Generatoren beschreibt den passiven Istzustand des Nano-Grids hinsichtlich seiner Spannungs- und Kapazitätswerte sowie die benötigten Gesamtzustände für das Schnittstellenmanagement. Die Ereignisse in diesen Generatoren sind in der Regel nicht steuerbar, können aber in der Realität durch Messwerterfassung beobachtet werden.

Folgende Tabelle gibt eine Übersicht über alle erstellten Streckengeneratoren mit einer kurzen Beschreibung und der entsprechenden Bezeichnung. Die einzelnen Generatoren und die darin enthaltenen Ereignisse werden jeweils in den nachfolgenden Kapiteln beschrieben. Die Ereignisse sind global eindeutig definiert.

Tabelle 13: Beschreibung der modellierten Streckengeneratoren

Typ	Bezeichnung	Beschreibung
aktiv	G_R	Verhalten der Betriebszustände der Komponente *DC/DC Regenerativ*
	G_N	Verhalten der Betriebszustände der Komponente *DC/DC Nicht-Regenerativ*
	G_S	Verhalten der Betriebszustände der Komponente *DC/DC Speicher*
	G_L	Verhalten der Betriebszustände der Komponente *DC/DC Last*
	$G_{directL}$	Verhalten der Betriebszustände des Lastschalters
	G_P	Verhalten der Betriebszustände der Komponente *AC/DC Öffentliches Netz* (der Index P leitet sich aus dem Englischen Public Grid ab)
	G_I	Verhalten der Betriebszustände der Komponente *DC/DC Schnittstelle*(der Index I leitet sich aus dem Englischen Public Grid ab)
passiv	G_{SOC}	Verhalten der SOC Zustände der angeschlossenen Speicherkomponente
	$G_{Grid,V}$	Verhalten der Spannungszustände des Verteilnetzes
	$G_{Grid,Status}$	Verhalten der Gesamtzustände des lokalen und externen Nano-Grids bzw. Verbundnetzes

Die jeweiligen Betriebszustände der einzelnen DC/DC-Konverter wurden in Kapitel 3.2 beschrieben.

5.1.1 Generatoren der Erzeugerkomponenten

Für **die regenerative Komponente** wird das logische Verhalten des Konverters *DC/DC Regenerativ* anhand des Generators G_R beschrieben. Der Generator besitzt folgende Ereignisse:

Tabelle 14: Beschreibung der Ereignisse des Generators G_R

Ereignisname	Steuerbarkeit	Beschreibung
r_source_idle	steuerbar	Steuert den Betriebszustand *Idle* an
r_source_cv	steuerbar	Steuert den Betriebszustand *Constant Voltage* an
r_source_cp	steuerbar	Steuert den Betriebszustand *Constant Power* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Die folgende Abbildung stellt das ungesteuerte Systemverhalten von G_R dar.

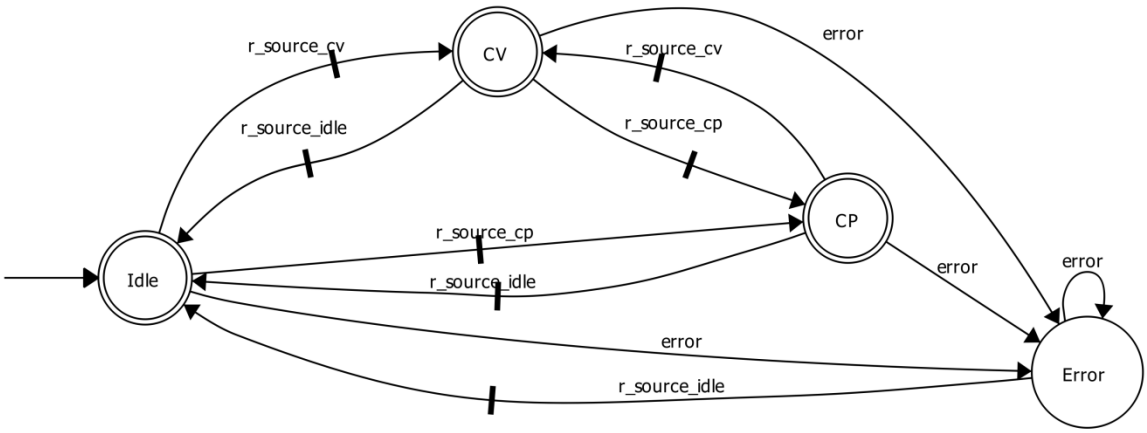

Abbildung 22:Generator G_R der Komponente *DC/DC Regenerativ*

Der Generator startet im Initialzustand *Idle*. Von diesem können alle weiteren Zustände erreicht werden. Das Modell nimmt an, dass die steuerbaren Ereignisse Steuerbits im Softwarecode darstellen, welche die entsprechenden Funktionen und damit die Betriebszustände ansteuern. Der Zustand *CP* aktiviert die MPPT-Regelung, der Zustand *CV* die Voltage-Droop Regelung.

Das Ereignis *error* stellt das einzige nicht steuerbare Ereignis dar, da dieses von extern, aktiviert durch den Anwender, auftritt. Der Zustand *Error* kann nur durch das Ereignis *r_source_idle* verlassen werden. Das Ereignis *r_source_idle* stellt somit eine Quittierung des Systems dar. Das Ereignis *error* wird als Schleife am Zustand *Error* angebracht, da immer ein Fehler global im ganzen System eingelesen wird. Alle Zustände außer der Zustand *Error* sind markiert, da diese einen erfolgreich erreichten Prozess repräsentieren.

Für **die nicht regenerative Komponente** wird das logische Verhalten des Konverters *DC/DC Nicht-Regenerativ* anhand des Generators G_N beschrieben. Folgende Ereignisse sind im Generator vorhanden.

Tabelle 15: : Beschreibung der Ereignisse des Generators G_N

Ereignisname	**Steuerbarkeit**	**Beschreibung**
n_source_idle	steuerbar	Steuert den Betriebszustand *Idle* an
n_source_cv	steuerbar	Steuert den Betriebszustand *Constant Voltage* an
n_source_cp	steuerbar	Steuert den Betriebszustand *Constant Power* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Das ungesteuerte Systemverhalten ist durch folgenden Automaten beschrieben.

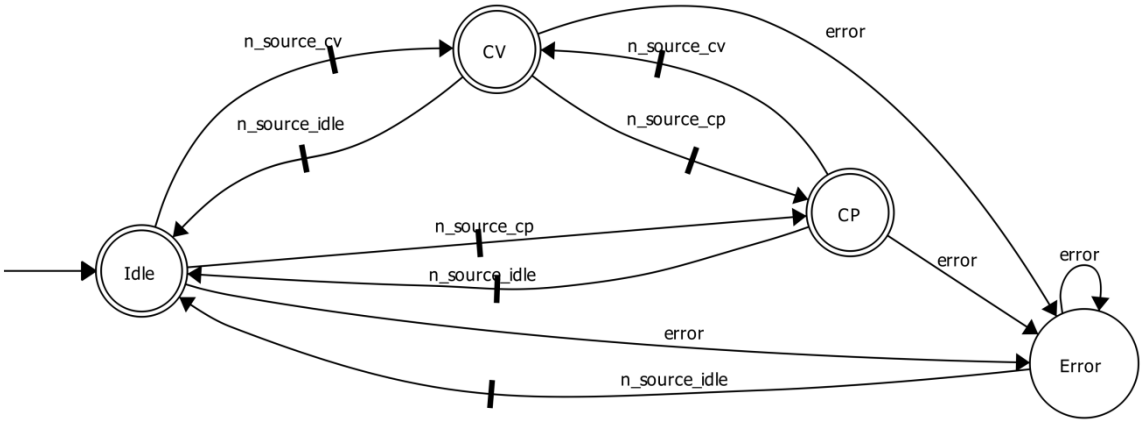

Abbildung 23: Generator G_N der Komponente *DC/DC Nicht-Regenerativ*

Das Verhalten der Komponente *DC/DC Nicht-Regenerativ* entsprichtt dem Verhalten der Komponente *DC/DC Regenerativ* und unterscheiden sich nur in der Bezeichnung der steuerbaren Ereignisse.

5.1.2 Generator der Speicherkomponente

Für die Speicherkomponente wird das logische Verhalten des Konverters *DC/DC Storage* anhand des Generators G_S beschrieben. In folgender Tabelle sind die modellierten Ereignisse beschriebenen

Tabelle 16: Beschreibung der Ereignisse des Generators G_S

Ereignisname	**Steuerbarkeit**	**Beschreibung**
storage_idle	steuerbar	Steuert den Betriebszustand *Idle* an
storage_chrg	steuerbar	Steuert den Betriebszustand *Charge* an
storage_dischrg	steuerbar	Steuert den Betriebszustand *Discharge* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Das ungesteuerte Systemverhalten ist durch den abgebildeten Automaten beschrieben.

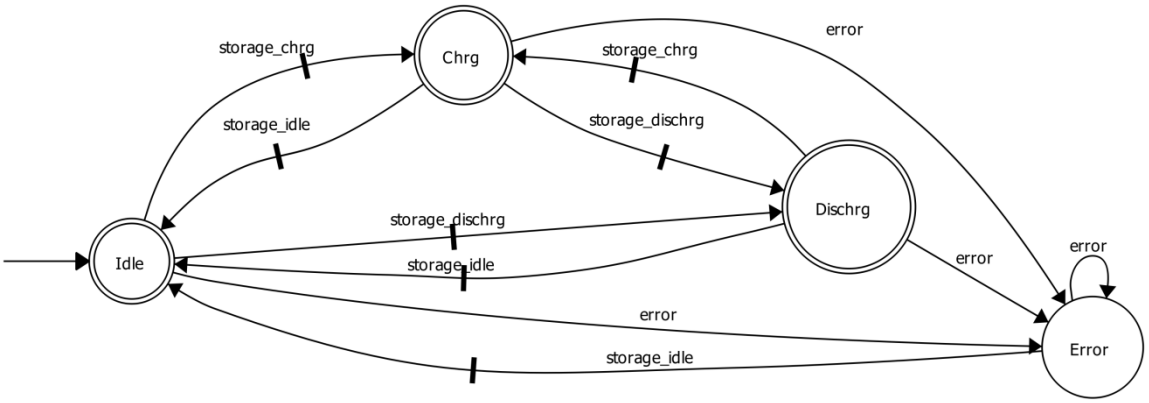

Abbildung 24: Generator G_S der Komponente *DC/DC Speicher*

Die Speicherkomponente startet im Zustand *Idle*. Aus diesem Zustand können alle weiteren Zustände erreicht werden. Über das Ereignis *storage_chrg* wechselt der Generator zum Zustand *Chrg*, welcher das Laden des angeschlossenen Speichers mit dem entsprechenden Ladealgorithmus aktiviert. Durch das Ereignis *storage_dischrg* wird der Zustand *Dischrg* erreicht. Dieser aktiviert das Entladen der Speicherkomponente. Der Zustand *Error* kann von allen Zuständen durch das Ereignis *error* erreicht werden und nur durch das Ereignis *storage_idle* wieder verlassen werden.

5.1.3 Generatoren der Lastkomponenten

Für die Lastkomponente wird das logische Verhalten des **Konverters *DC/DC Last*** anhand des Generators G_L beschrieben Die modellierten Ereignisse des Generators G_L werden in folgender Tabelle beschrieben.

Tabelle 17: Beschreibung der Ereignisse des Generators G_L

Ereignisname	**Steuerbarkeit**	**Beschreibung**
dc_load_off	steuerbar	Steuert den Betriebszustand *Off* an
dc_load_on	steuerbar	Steuert den Betriebszustand *On* an
dc_load_on_limit	steuerbar	Steuert den Betriebszustand *Limit* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Der abgebildete Automat in Abbildung 25 stellt das ungesteuerte Systemverhalten der Streckenkomponente *DC/DC Last* dar.

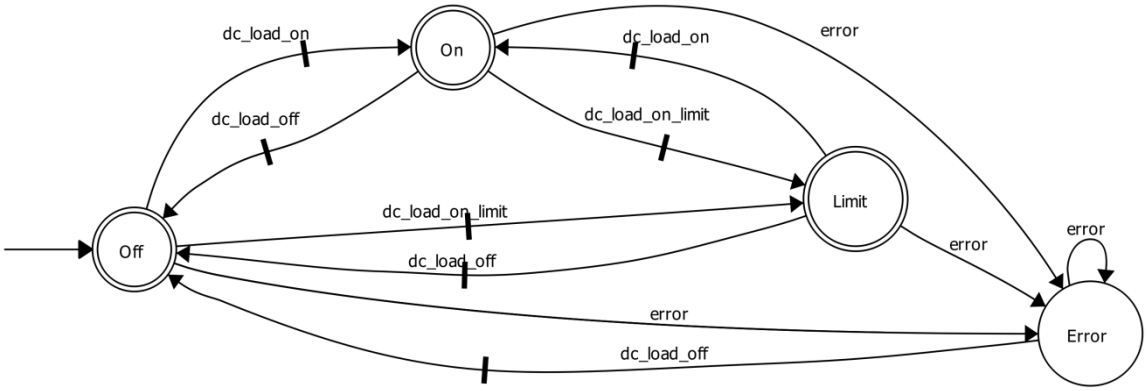

Abbildung 25: Generator G_L der Komponente *DC/DC Last*

Der Automat startet im Zustand *Off*. Von dort aus können alle weiteren Zustände erreicht werden. Über das steuerbare Ereignis *dc_load_on* wird der der Zustand *On* erreicht, welcher einen Stromfluss zur angeschlossenen Last zulässt. Es wird angenommen, dass das Ereignis *dc_load_on* aus einer übergeordneten Anfragebewertung generiert wird. Diese Anfrage kann entweder über einen Kommunikationsbus, falls dieser in der Steuerungsarchitektur vorgesehen ist, erfolgen oder durch einen Hardwareschalter, den der Anwender betätigen kann. Alternativ wird das Ereignis *load_on_limit* generiert, was den Zustand *Limit* aktiviert, in welchem die Lastleistung begrenzt wird. Das Anfrageverhalten wird nicht modelliert, da es für die geforderte Steuerungsaufgabe irrelevant ist.

Das Erreichen und Verlassen des Zustands *Error* ist analog der oben aufgeführten Generatormodelle.

Um das ungesteuerte Verhalten **des Lastschalters** abzubilden wird der Generator $G_{directL}$ modelliert. Die vorhandenen Ereignisse sind wie folgt beschrieben:

Tabelle 18: Beschreibung der Ereignisse des Generators $G_{directL}$

Ereignisname	**Steuerbarkeit**	**Beschreibung**
load_off	steuerbar	Steuert den Betriebszustand *Off* an
load_on	steuerbar	Steuert den Betriebszustand *On* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Das ungesteuerte Systemverhalten ist durch das folgende Automatenmodell abgebildet.

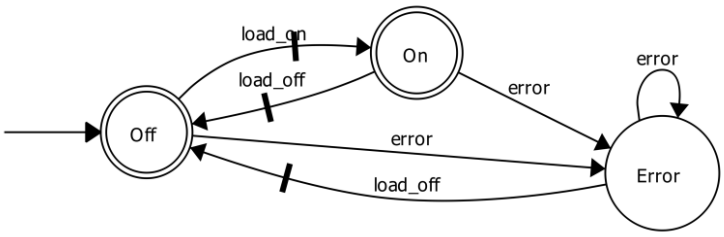

Abbildung 26: Generator $G_{directL}$ des Lastschalters

Der Automat befindet sich zu Beginn im Zustand *Off*. Von dort aus kann über die steuerbaren Ereignisse *laod_on* und *load_off* zwischen den Zuständen *On* und *Off* gewechselt werden. Der Zustand *Error* ist für das Eintreten des Fehlereignisses *error* modelliert, damit auch die Lastschalterkomponente sich in einem definierten Fehlerzustand befindet. Ein einfacher logischer Schalter besitzt eigentlich keinen Fehlerzustand. Die softwaretechnische Ansteuerung des Schalters besitzt jedoch diesen und ist in diesem Generator durch den genannten Zustand berücksichtigt. Durch das Ereignis *load_off* kann der Zustand *Error* verlassen werden.

5.1.4 Generator der Schnittstellenkomponenten

Das logische Verhalten **der Schnittstellenkomponente *DC/DC Schnittstelle*** wird anhand des Generators G_I beschrieben. Folgende Ereignisse sind im Generator vorhanden.

Tabelle 18: Beschreibung der Ereignisse des Generators G_I

Ereignisname	**Steuerbarkeit**	**Beschreibung**
interface_idle	steuerbar	Steuert den Betriebszustand *Idle* an
power_int	steuerbar	Steuert den Betriebszustand *Power internal* an
power_ext	steuerbar	Steuert den Betriebszustand *Power external* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

Das ungesteuerte Systemverhalten ist durch folgendes Automatenmodell beschrieben.

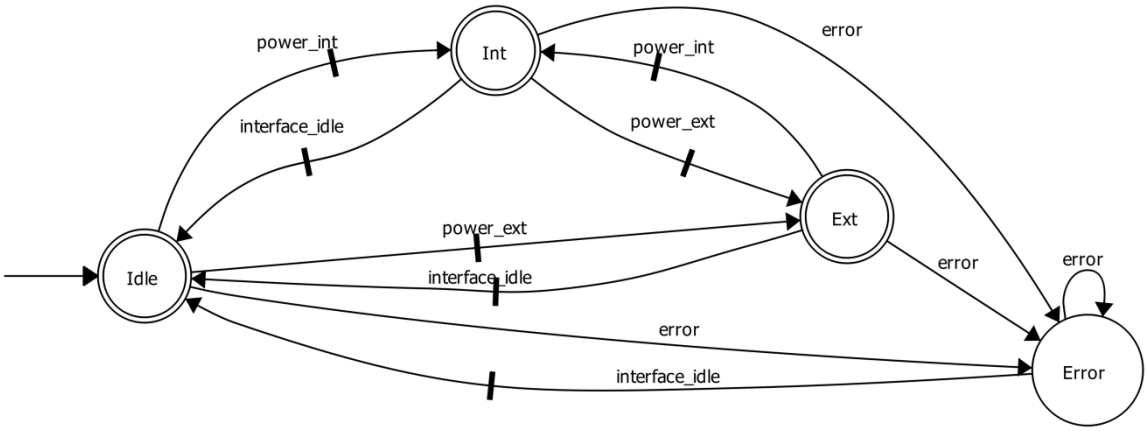

Abbildung 27: Generator G_I der Komponente *DC/DC Schnittstelle*

Der Generator startet im Zustand *Idle*. Von diesem können alle Betriebszustände erreicht werden. Im Zustand *Int*, welcher über das Ereignis *power_int* erreicht werden kann, wird der Stromfluss vom angeschlossenen externen System in das lokale Nano-Grid angesteuert. Das externe System kann in diesem Fall ein weiteres Nano-Grid sein oder das Verbundnetz. Das Ereignis *powert_ext* wechselt zum Zustand *Ext*, durch welchen der Betriebszustand *Power external* der Komponente *DC/DC Schnittstelle* aktiviert. Das Ereignis *error* kann in jedem Zustand auftreten. Mit dem Ereignis *interface_idle* wird der Zustand Error verlassen.

Das Verhalten **der Komponente *AC/DC Öffentliches Netz***, welche für die Kopplung mit einem öffentlichen Netz genutzt wird, wird anhand des Generators G_P beschrieben. Folgende Ereignisse treten in diesem Generator auf.

Tabelle 19: Beschreibung der Ereignisse des Generators G_P

Ereignisname	**Steuerbarkeit**	**Beschreibung**
public_idle	steuerbar	Steuert den Betriebszustand *Idle* an
to_grid	steuerbar	Steuert den Betriebszustand *Power from grid* an
from_grid	steuerbar	Steuert den Betriebszustand *Power from grid* an
error	nicht steuerbar	Führt das System in den Fehlerzustand *Error*

In Abbildung 28 bildet das ungesteuerte Systemverhalten des Generators G_P ab.

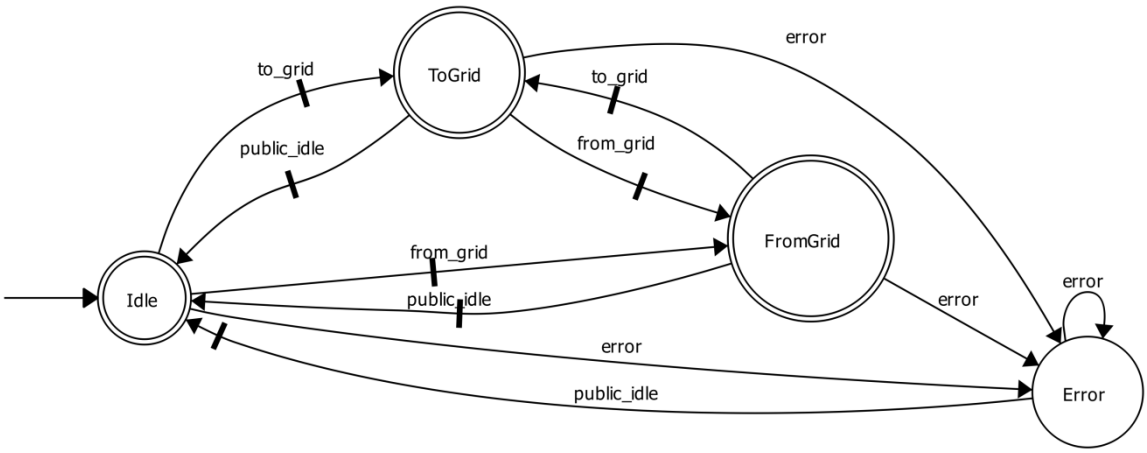

Abbildung 28: Generator G_P der Komponente *AC/DC Öffentliches Netz*

Der Generator startet im Zustand *Idle*. Von diesem können alle Betriebszustände erreicht werden. Im Zustand *ToGrid*, welcher über das Ereignis *to_grid* erreicht werden kann, wird der Stromfluss vom lokalen Nano-Grid zum Öffentlichen Netz angesteuert. Das Ereignis *from_grid* führt den Generator zum Zustand *FromGrid* über Der Betriebszustand *Power external* der Komponente *AC/DC Öffentliches Netz* wird aktiviert. Das Ereignis *error* kann in jedem Zustand auftreten. Mit dem Ereignis *public_idle* wird der Zustand Error verlassen. Über das Ereignis *public_idle* kann von jedem Zustand aus der Initialzustand erreicht werden.

5.1.5 Generator des Speicherladezustands

Der Ladezustand der angeschlossenen Speicherkomponente wird anhand des Generators G_{SOC} beschrieben. Im Gegensatz zu den zuvor beschriebenen Generatoren, besitzt dieser Generator keine steuerbaren Ereignisse. Das Modell beschreibt somit kein Softwaremodell, in welchem die Ereignisse Steuerbits zum Ansteuern von Betriebszuständen darstellen. Die modellierten Ereignisse beschreiben das Erreichen von Schwellwertgrenzen des SOCs, welche durch die Speicherkomponente berechnet werden. Die Berechnung des entsprechenden SOCs ist für die Modellbetrachtung nicht relevant. Folgende SOC Kurve stellt exemplarisch die Ereignisgenerierung der modellierten Ereignisse dar.

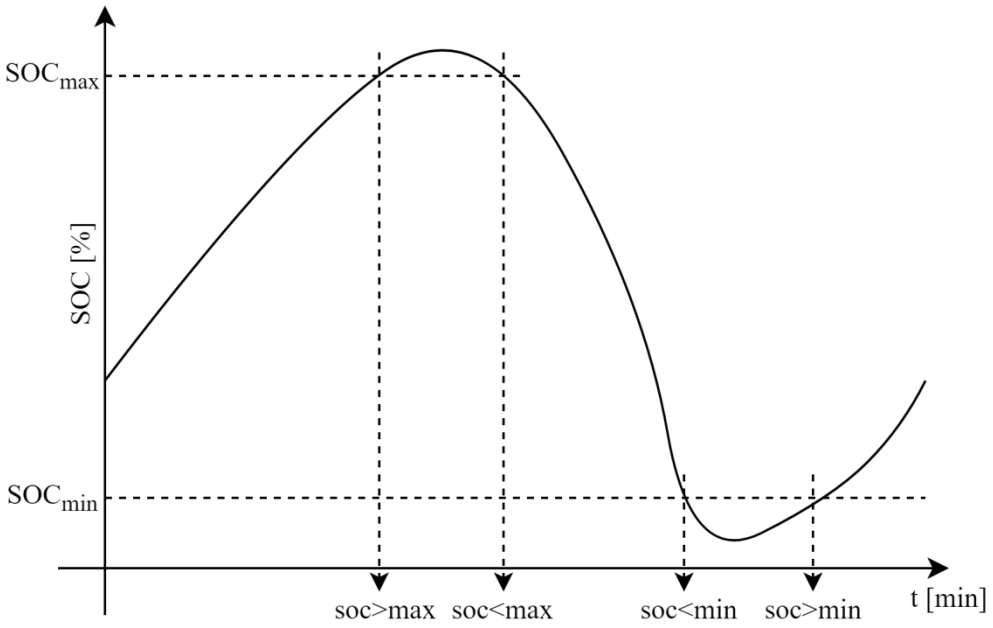

Abbildung 29: Generierung der Schwellwertereignisse des Generators G_{SOC}

Damit sind die Ereignisse des Generators G_{SOC} folgendermaßen beschrieben:

Tabelle 20: Beschreibung der Ereignisse des Generators G_{SOC}

Ereignisname	**Steuerbarkeit**	**Beschreibung**
soc>max	nicht steuerbar	Schwellwert SOC_{max} wird überschritten
soc<max	nicht steuerbar	Schwellwert SOC_{max} wird unterschritten
soc<min	nicht steuerbar	Schwellwert SOC_{min} wird unterschritten
soc>min	nicht steuerbar	Schwellwert SOC_{min} wird überschritten

Das Systemverhalten für den Ladezustand wird mit folgendem Automaten beschrieben.

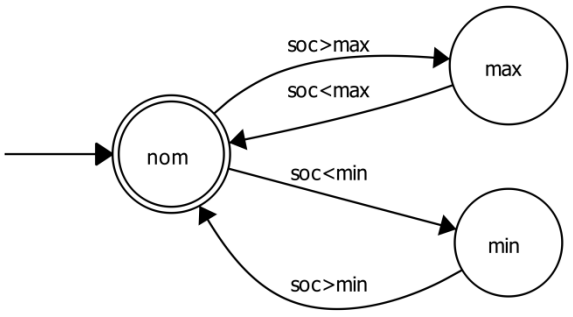

Abbildung 30: Generator G_{SOC} des SOC-Zustands der Speicherkomponente

Es wird angenommen, dass der Ladezustand beim Starten des Nano-Grids sich innerhalb der gewünschten Grenzwerte befindet, was durch den Zustand *nom* (für nominal) repräsentiert wird. Durch Eintreten des Ereignisses *soc>max*, wechselt der Generator in den Zustand *max*. Es wird angenommen, dass kein direkter Sprung des SOCs vom oberen Grenzwert zum unteren erfolgt, daher kann der Zustand *max* nur über das Ereignis *soc<max* über den Zustand *nom* und durch das Eintreten des Ereignisses *soc<min* zum Zustand *min* gelangen. Entsprechend kann dieser nur durch Eintreten von Ereignis *soc>min* den Zustand *nom* erreichen. Es wird nur der Zustand *nom* markiert, da dieser einen gewünschten Zustand des Systems darstellt.

5.1.6 Generator des Verteilnetzspannungszustands

Für das geforderte Leistungsmanagement ist die Information über den Leistungszustand im Nano-Grid notwendig. Diese ergibt sich aus der Leistungsbilanz zwischen Erzeugern und Quellen. Für den Steuerungsentwurf wird die DBS Methode angewendet. Somit muss der Zustand der Verteilnetzspannung bekannt sein. Dazu wird der Generator $G_{Grid,V}$ modelliert. Analog zum SOC werden die modellierten Ereignisse durch Erreichen von definierten Schwellwerten der Verteilnetzspannung generiert. Folgende Kurve zeigt den Spannungsverlauf $G_{Grid,V}$ in Abhängigkeit des Laststroms I_L und den definierten Schwellwerten V_i mit $i = [0 \dots 5]$.

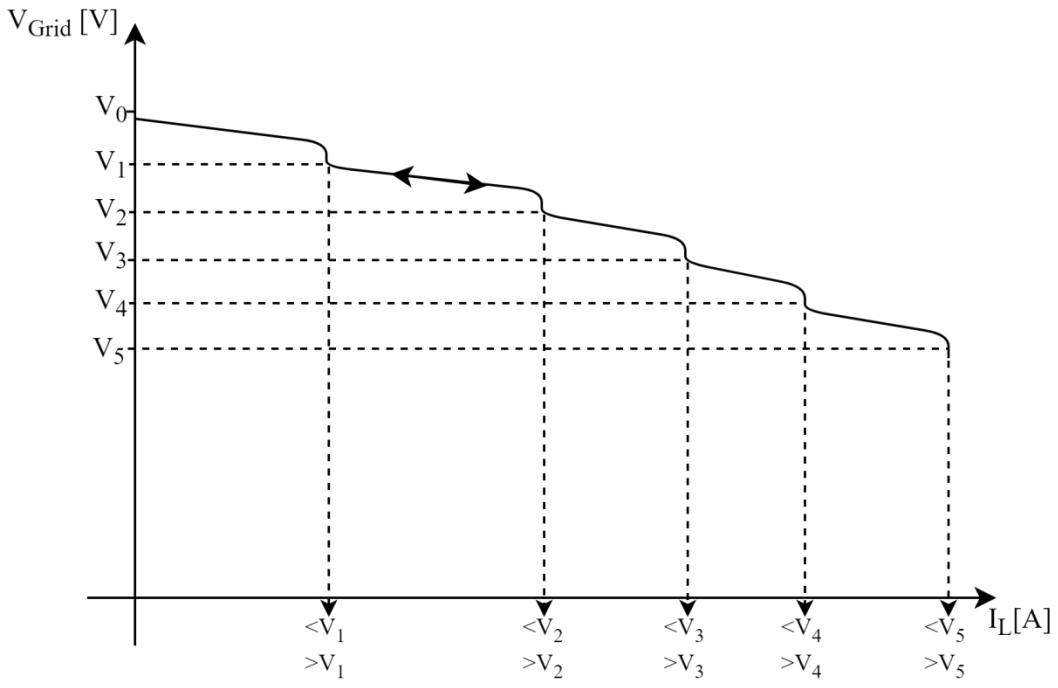

Abbildung 31: Generierung der Schwellwertereignisse des Generators $G_{Grid,V}$

Für jedes Über- und Unterschreiten eines Schwellwertes wird das entsprechende Ereignis $<V_i$ bei Unterschreiten und das Ereignis $>Vi$ bei Überschreiten eines Schwellwertes V_i mit $i = [0 \dots 5]$ generiert. Der Schwellwert V_0 dient als obere Spannungsgrenze. Der Schwellwert V_5 dient als untere Spannungsgrenze.

In der folgenden Abbildung ist das Spannungsverhalten als Automatenmodell formal beschrieben.

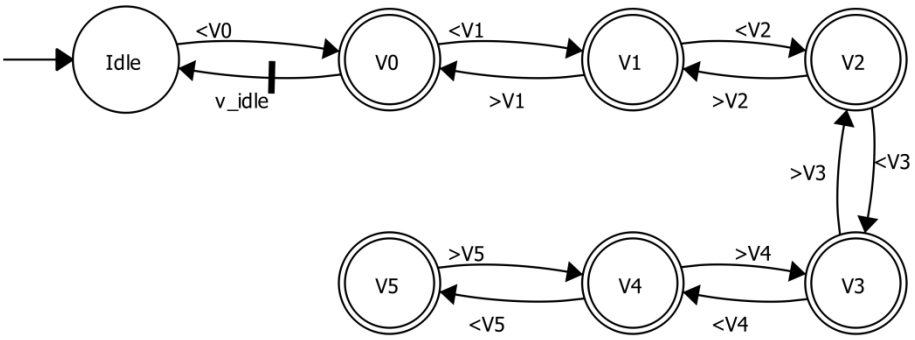

Abbildung 32: Generator $G_{Grid,V}$ des Spannungszustands des Verteilnetzes

Der Initialzustand *Idle* beschreibt den Handbetrieb, in welchem durch das nicht betrachtete Aktivieren der Messwerterfassung die Information des Spannungszustands eingelesen wird. Es wird angenommen, dass das System ohne angeschlossene Lasten startet und damit die Spannung unterhalb des Spannungsschwellwertes V_0 und oberhalb des ersten Schwellwertes V_1 liegt. Da der Spannungswert eine kontinuierliche Größe darstellt, werden die weiteren Schwellwerte sequentiell durchlaufen. Damit wird angenommen, dass keine Spannungssprünge über zwei Schwellwerte hinweg auftreten können. Nach jedem Schwellwertübergang wird der erreichte Zustand markiert.

Um wieder in den Zustand *Idle* zurückzukehren wird das steuerbare Ereignis *v_idle* modelliert, welches einen externen Eingriff durch den Anwender oder durch ein Steuerbit in der Software darstellt.

5.1.7 Generator des Gesamtzustands der gekoppelten Nano-Grids

Um die Kopplung des Nano-Grids mit externen Systemen zu steuern, muss der Gesamtzustand des lokalen Nano-Grids und des externen Systems bekannt sein. Hierfür wird der Generator $G_{Grid,Status}$ modelliert. Die Ereignisse, die einen Zustandswechsel anstoßen, werden durch Kombination der einzelnen Attribute aus Tabelle 9 generiert. Somit existieren insgesamt neun Ereignisse, die den Gesamtzustand beider Systeme darstellen. Die gegebene Kombinatorik ist in folgender Tabelle dargestellt.

Tabelle 21: Beschreibung der Ereignisgenerierung für $G_{Grid,Status}$

Ereignisname	**Beschreibung: Konjunktion aus den Systemzuständen von**	
	Lokales Nano-Grid	**Externes System**
sat_sat	saturated	saturated
sat_self	saturated	self-sufficient
sat_def	saturated	deficient
self_sat	self-sufficient	saturated
self_self	self-sufficient	self-sufficient
self_def	self-sufficient	deficient
def_sat	deficient	saturated
def_self	deficient	self-sufficient
def_def	deficient	deficient

Zusätzlich wird ein Ereignis *calc* (für calculte) modelliert. Dieses wird verwendet um einen Zustandswechsel von einen der beiden Systeme zu erkennen. Folgender Programmablaufplan stellt die Ereignisgenerierung für *calc* dar.

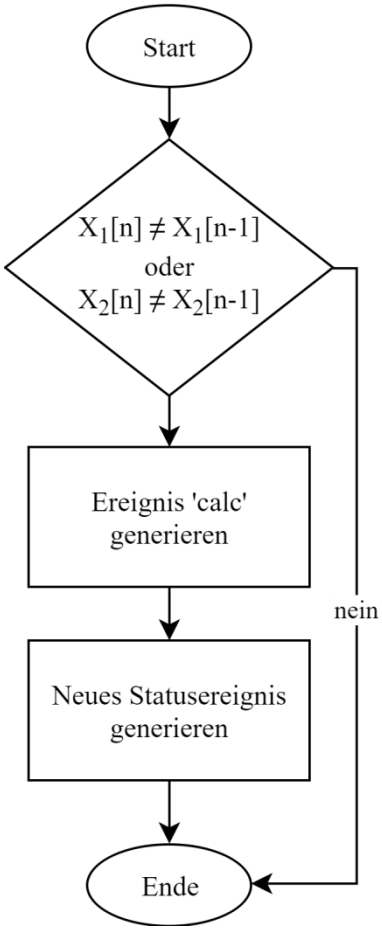

Abbildung 33: Programmablauf zur Generierung des Ereignisses *calc*

Der Zustand $X_1[n]$ stellt den Zustand des lokalen Nano-Grids zum Berechnungszeitpunkt n dar. Der Zustand $X_2[n]$ stellt den Zustand des externen Systems zum Berechnungszeitpunkt dar n. Sobald sich einer der Zustände ändert, wird das Ereignis *calc* generiert. Die Zustandsänderung wird durch den Vergleich des aktuellen Zustands $X_i[n]$ mit dem vorherigen Zustand $X_i[n-1]$ mit $i = \{1,2\}$ erkannt. Anschließend wird durch die oben beschriebene Konjunktion das neue Stautsereignis generiert.

Mit den beschriebenen Modellannahmen wird das Systemverhalten des Generators $G_{Grid,Status}$ wie folgt modelliert.

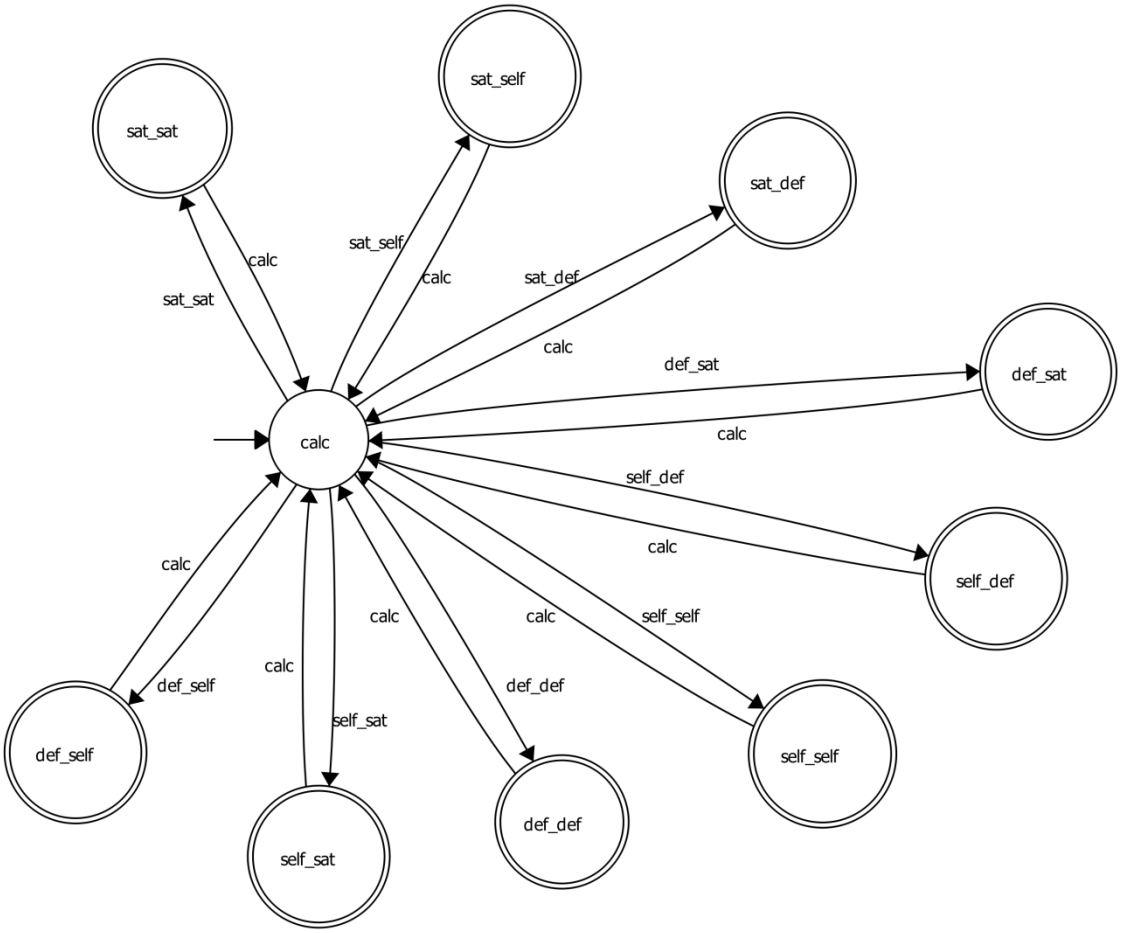

Abbildung 34: Generator $G_{Grid,Status}$ des Gesamtzustands der gekoppelten Nano-Grid-Systeme

Der Generator startet im Zustand *calc*. Je nach Kombination der lokalen **Nano-Grid**zustände wir das entsprechende Ereignis generiert und der Automat wechselt in den entsprechenden Zustand. Alle von *calc* erreichten Zustände sind markiert und können nur durch das Ereignis *calc* wieder verlassen werden. Ein Zustandswechsel zwischen den Zuständen ist somit nur über den Zustand *calc* gegeben.

5.2 Modell der formalen Spezifikationen

Die formalen Modelle der Spezifikation leitet sich aus den in Kapitel 3.3 vorgestellten informell beschriebenen Steuerstrategien zum Leistungs-, Speicher- und Sicherheitsmanagement ab. Die geforderte Steuerungsaufgaben werden so modularisiert, dass sowohl eine zentrale, als auch eine verteilte Implementierung erfolgen kann. Gibt eine Übersicht der modellierten Spezifikationen

Tabelle 22: Beschreibung der modellierten Spezifikationen

Bezeichnung	Beschreibung
K_R	Spezifikation für das Erzeugermanagement der Komponente *DC/DC Regenerativ*
K_N	Spezifikation für das Erzeugermanagement der Komponente *DC/DC Nicht-Regenerativ*
K_S	Spezifikation für das Erzeugermanagement der Komponente *DC/DC Speicher*
K_L	Spezifikation für das Lastmanagement der Komponente *DC/DC Last*
$K_{directL}$	Spezifikation für das Lastmanagement des Lastschalters
K_P	Spezifikation für das Schnittstellenmanagement der Komponente *AC/DC Öffentliches Netz*
K_I	Spezifikation für das Schnittstellenmanagement der Komponente *DC/DC Schnittstelle*
K_{SOC}	Spezifikation für das Speichermanagement
K_{Safety}	Spezifikation für das Sicherheitsmanagement

Die Spezifikationen für das Leistungsmanagement werden für jeden DC/DC-Konverter sowie für den Lastschalter erstellt. Dabei wird zwischen Erzeuger-, Last- und Schnittstellenmanagement unterschieden.

Um eine verteilte Implementierung mit Hilfe des lokal-modularen Entwurfsansatzes zu erreichen, wird angenommen, dass die Streckenmodelle der aktiven Streckenkomponenten G_i mit $i = \{R, S, N, L, directL, I, P\}$ asynchron zueinander laufen. Daher wird das Ereignis *error* aus den jeweiligen Ereignisalphabeten Σ_i mit $i = \{R, S, N, L, directL, I, P\}$ heraus modelliert. Somit wird dieses auch nicht in den Spezifikationsmodellen berücksichtigt. Die angepassten Streckengeneratoren werden jetzt mit neu $G_{i,P}$ und den Ereignisalphabeten $\Sigma_{i,P}$ mit $i = \{R, S, N, L, directL, I, P\}$ bezeichnet. Die Indexerweiterung P steht für den englischen Ausdruck Powermanagement (auf Deutsch: Leistungsmanagement). Das Ereignis *error* wird separat durch das Sicherheitsmanagement und der Spezifikation K_{Safety} berücksichtigt.

Die generelle Modellierungsstrategie ist die steuerbaren Ereignisse der aktiven Streckenkomponenten in Abhängigkeit der nicht steuerbaren Ereignisse der passiven Streckenkomponenten einzuschränken.

5.2.1 Spezifikationen des Erzeugermanagements

Beginnend mit dem Erzeugermanagement wird für die regenerative Komponente, die nicht-regenerative Komponente und Speicherkomponente ein Spezifikationsmodell erstellt, welches die Betriebszustandswechsel der Komponenten in Abhängigkeit zum Spannungszustand des Verteilnetzes beschränken. Es wird für jede Komponente eine separate Spezifikation erstellt, wodurch die einzelnen Komponenten unabhängig voneinander arbeiten.

Für das **Erzeugermanagement der Komponente *DC/DC Regenerativ*** wird die Spezifikation K_R modelliert. Das Spezifikationsmodell ist in der folgenden Abbildung dargestellt.

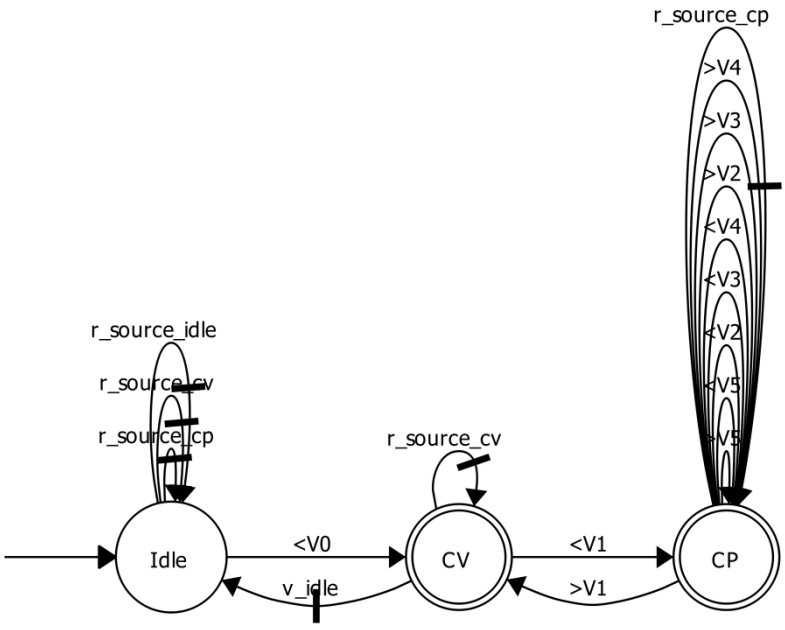

Abbildung 35: Generator der Spezifikation K_R für die Komponente *DC/DC Regenerativ*

Im Zustand *Idle* wird jedes Ereignis zugelassen. Es wird angenommen, dass die Messwerterfassung noch nicht aktiviert ist und damit noch keine Ereignisse aus dem Generator $G_{Grid,V}$ generiert werden. Damit entspricht der Zustand dem Handbetrieb, in welchem alle Betriebszustände erlaubt sind. Sobald das Ereignis *<V0* eintritt, wechselt der Generator in den Zustand *CV*. Es wird nur noch das steuerbare Ereignis *r_source_cv* erlaubt, welches die Voltage-Droop Regelung aktiviert. Dies entspricht Zustand 1 aus der Steuerregel in Tabelle 7.

Sobald der nächste Schwellwert unterschritten wird und damit das Ereignis *<V1* eintritt, wechselt der Generator in den Zustand *CP*, in welchem nur das steuerbare Ereignis *r_source_cp* zugelassen wird. Damit wird die MPPT-Regelung aktiviert. Diese soll für alle weiteren Schwellwert über V_1 aktiv blieben, weshalb im Zustand *CP* die weiteren nicht steuerbaren Schwellwerte als Schlingen modelliert sind. Erst wenn die Spannung oberhalb von V_1 ansteigt und damit das Ereignis *>V1* auftritt, wechselt der Generator in den Zustand *CV* zurück. Durch das steuerbare Ereignis *v_idle* kann das System wieder in den Zustand *Idle* überführt werden.

Für das **Erzeugermanagement der Komponente *DC/DC Regenerativ*** wird die Spezifikation K_N modelliert. Sie unterscheidet sich zur Spezifikation K_R in den auftretenden Schwellwerten, die die Betriebszustandsauswahl beschränken.

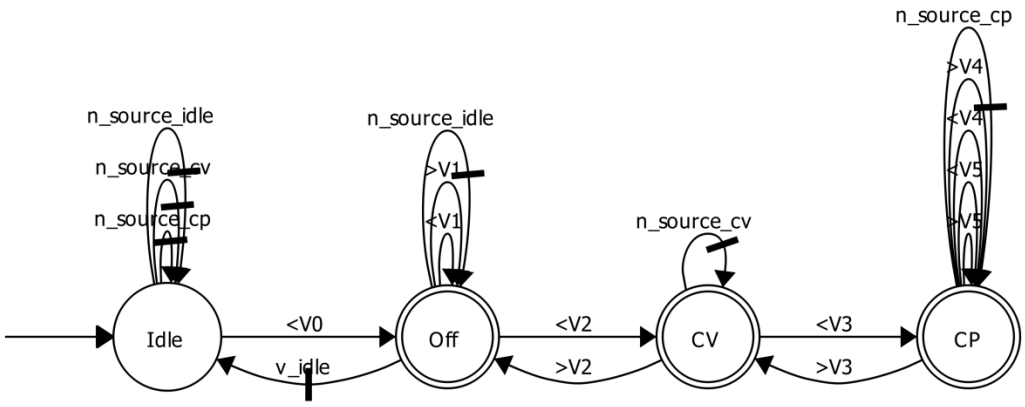

Abbildung 36: Generator der Spezifikation K_N für die Komponente *DC/DC Nicht-Regenerativ*

Der Zustand *Idle* repräsentiert den Handbetrieb. Sobald die Messwerterfassung startet und der erste Schwellwert V_0 unterschritten wird, wird die Komponente *DC/DC Nicht-Regenerativ* auf ihr steuerbare Ereignis *n_source_idle* beschränkt und darf noch nicht eingeschaltet werden. Erst nach Unterschreiten des Schwellwertes V_2 wird der Zustand *CV* aktiv, in welchem lediglich das Ereignis *n_source_cv* erlaubt, welches unterlagert die Voltage-Droop Regelung aktiviert.

Fällt die Spannung weiter und unterschreitet den Schwellt V_3, generiert die ungesteuerte Strecke $G_{Grid,V}$ das Ereignis *<V3*. Der Spezifikationsgenerator wechselt in den Zustand *CP* und erlaubt nur noch das steuerbare Ereignis *n_source_cp*, welches den Betriebszustand *Constant Power* aktiviert. Der Betriebszustand soll solang aktiv bleiben bis der Spannungswert wieder oberhalb des Schwellwerts V_3 liegt. Die restlichen Schwellwertereignisse werden in Form von Schlinge an den Zustand *CP* modelliert. Steigt die Verteilnetzspannung wieder an und überschreitet die Schwellwerte V_3 und V_2, findet ein

entsprechender Zustandswechsel zurück statt. In den Handbetriebszustand *Idle*, kann die Komponente erst aus dem Zustand *Off* durch das steuerbare Ereignis *v_idle* gelangen.

Das **Erzeugermanagement der Komponente DC/DC Speicher** wird anhand der Spezifikation K_S abgebildet. In folgender Abbildung ist das Modell der Spezifikation K_S abgebildet:

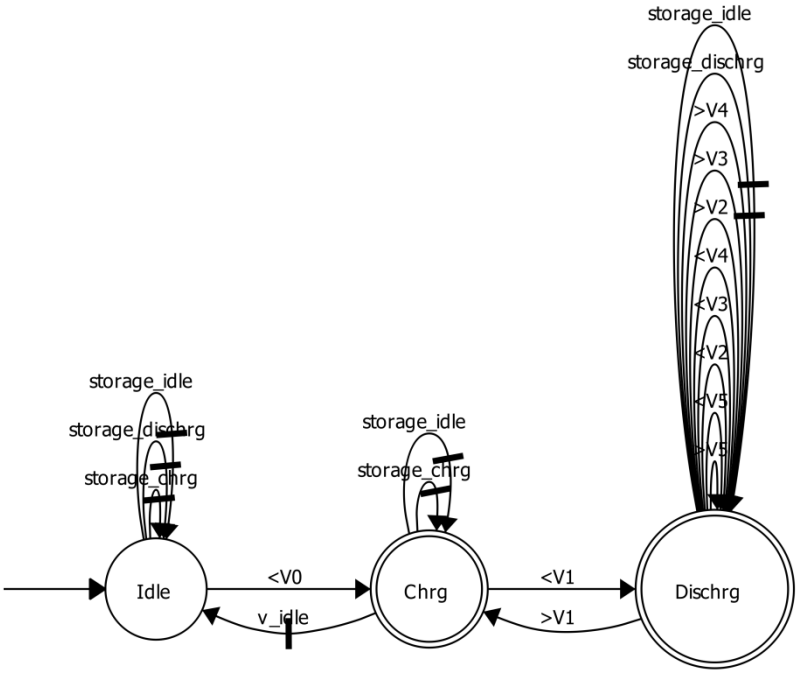

Abbildung 37: Generator der Spezifikation K_S für die Komponente *DC/DC Speicher*

Der Zustand *Idle* entspricht dem Handbetrieb. Nach Aktivierung der Messwerterfassung und dem Unterschreiten der Spannungsschwelle von V_0 geht der Generator in den Zustand *Chrg* über, in welchem die Ereignisse *storage_chrg* und *storage_idle* erlaubt werden. Damit kann der Speicher geladen werden. Tritt das nicht steuerbare Ereignis *<V1* auf, wechselt der Generator in den Zustand *Dischrg* und erlaubt nur noch das Ereignis *storage_dischrg*, welches das Entladen der Speichers erlaubt und das Ereignis *storage_idle*. Die Einschränkung den Speicher nur noch entladen zu können, soll für die weiteren Schwellwerte unterhalb von V_1 beibehalten werden, was durch die Modellierung von Schlingen am Zustand *Dischrg* erfolgt. Ein Verlassen des Zustands *Dischrg* wird durch ein Überschreiten des Schwellwerts V_1. möglich. Das Ereignis *storage_idle* wird an jedem Zustand erlaubt, um die Spezifikation des Sicherheitsmanagements nicht zu verletzen, im Fall, dass der Speicher seine SOC Grenzwerte erreicht.

5.2.2 Spezifikationen des Lastmanagements

Die Spezifikationen für das Erzeugermanagement schränken analog dem Erzeugermanagement die Betriebszustandswechsel der Lastkomponente *DC/DC Last* und des Lastschalters ein. Es wird jeweils eine Spezifikation modelliert.

Für das **Lastmanagement der Komponente *DC/DC Last*** wird die Spezifikation K_L modelliert, deren Generatormodell in folgender Abbildung dargestellt ist.

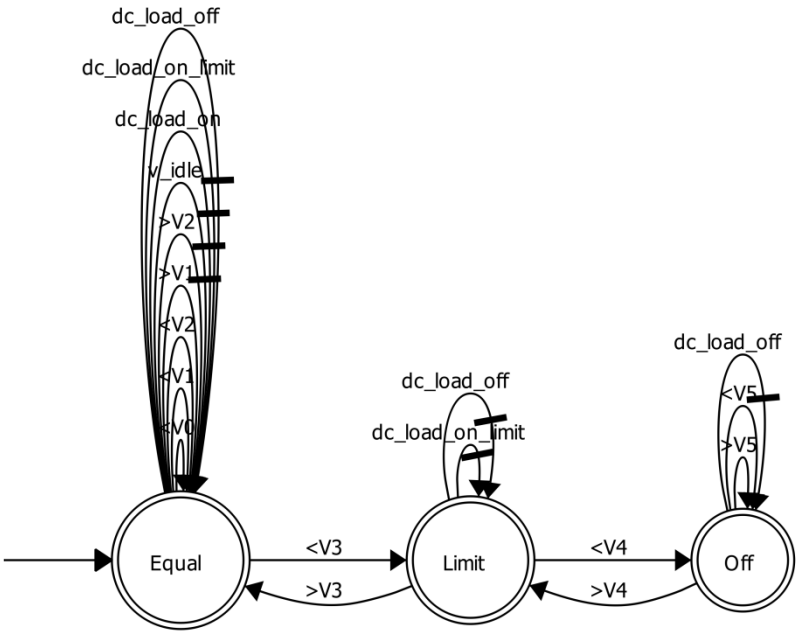

Abbildung 38: Generator der Spezifikation K_L für die Komponente *DC/DC Last*

Der Generator startet im Zustand *Equal*, in dem jedes steuerbare Ereignis der Komponente *DC/DC Last* erlaubt wird. Sobald das das Ereignis *<V3* eintritt, wechselt der Generator in den Zustand *Limit*. Das Ereignis signalisiert, dass der Spannungswert unterhalb des Schwellwerts V_3 liegt und die Erzeugerkomponenten nicht genügend Leistung für die Lasten liefern können. Im Zustand *Limit* wird somit das Ereignis *dc_load_on* verhindert. Die Komponente kann ausgeschaltet werden oder die abgegebene Leistung begrenzen.

Sinkt die Spannung weiter und unterschreitet den Schwellwert V_4, tritt das Ereignis *<V4* auf, welches einen Zustandswechsel zum Zustand *Off* herbeiführt. In diesem wird nur noch das steuerbare Ereignis *dc_load_off* erlaubt wird. Die weiteren Schwellwerte, in denen die Lastkomponente ausgeschaltet bleiben soll, werden mit Schlingen am Zustand *Off* modelliert.

Für **das Lastmanagement des Lastschalters** wird die Spezifikation $K_{directL}$ modelliert, welche eine analoge Modellstruktur zu den vorher beschriebenen Spezifikationen aufweist und in folgender Abbildung dargestellt ist.

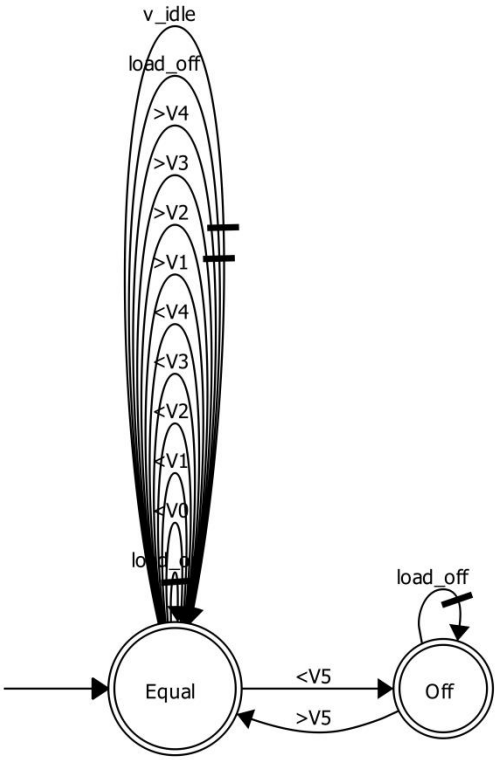

Abbildung 39: Generator der Spezifikation $K_{directL}$ für den Lastschalter

Der Initialzustand *Equal* erlaubt alle steuerbaren Ereignisse. Ein Wechsel zum Zustand *Off* findet erst nach Auftreten des Ereignisses *<V5* statt. Hier wird nur noch das steuerbare Ereignis *load_off* erlaubt, um die Last abzuschalten. Liegt die Spannung oberhalb des Schwellwerts V_5, so wechselt der Generator wieder zum Zustand *Equal*.

5.2.1 Spezifikationen des Schnittstellenmanagements

Die formale Spezifikation für das Schnittstellenmanagement des Nano-Grids umfasst die Komponenten *DC/DC Schnittstelle* und *AC/DC Öffentliches Netz*. Die Komponenten werden, anders als die Erzeuger- und Lastkomponenten, nicht in Abhängigkeit der Verteilnetzspannung, sondern in Abhängigkeit des Gesamtzustands der gekoppelten Nano-Gridsysteme in ihrer steuerbaren Ereignisgenerierung beschränkt. Die erlaubten Ereignisse werden von der Steuerstrategie aus Tabelle 10 abgeleitet. Die Spezifikation für das Schnittstellenmanagement läuft damit unabhängig vom Erzeuger- und Lastmanagement ab. Alternativ kann auch das DBS mit definierten Schwellwerten für das Steuern der

Schnittstellenkomponenten herangezogen werden. In dieser Modellierung wird das separat modellierte Streckenmodell $G_{Grid,Status}$ herangezogen.

Das **Schnittstellenmanagement für die Komponente DC/DC Schnittstellen** ist durch die Spezifikation K_I beschrieben und ist durch den folgenden abgebildeten Generator modelliert.

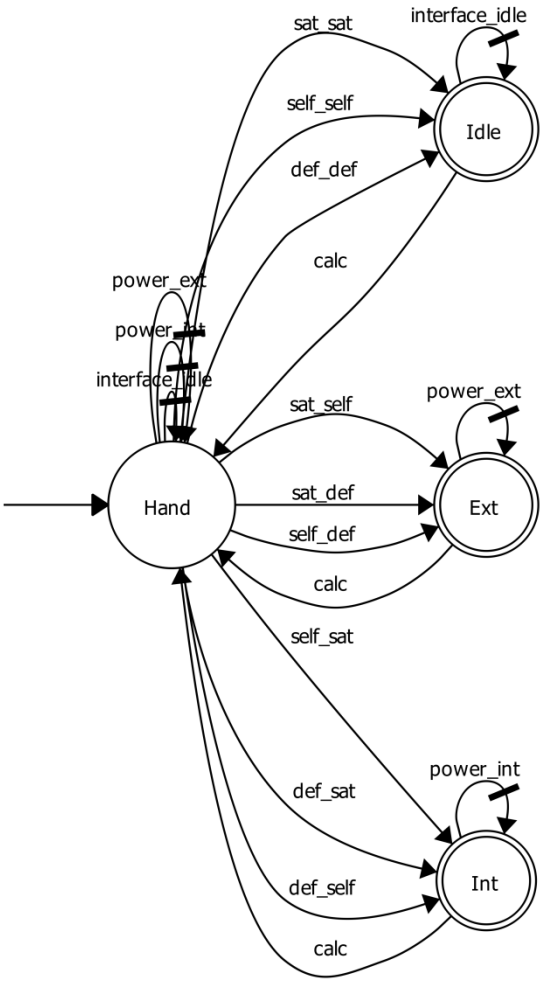

Abbildung 40: Generator der Spezifikation K_I für die Komponente *DC/DC Schnittstelle*

Auch in dieser Spezifikation wird jedes steuerbare Ereignis im Initialzustand erlaubt. Sobald das Statusereignis der gekoppelten Nano-Grids generiert wird, wechselt der Automat in einen der Zustände *Int*, *Ext* oder *Idle*. Der Zustandswechsel findet nach der Steuervorschrift aus Tabelle 10 statt. Im Zustand *Int* wird nur das Ereignis *power_int* erlaubt. Im Zustand Ext wird nur das Erlaubnis *power_ext* erlaubt und im Zustand *Idle* wird nur der Zustand *interface_idle* erlaubt.

Damit wird der Generator $G_{I,P}$ beispielsweise nach dem Auftreten des Ereignisses *sat_self* auf das steuerbare Ereignis *power_ext* beschränkt, welches den Betriebszustand *Power to extern* unterlagert aktiviert und damit die Komponente *DC/DC Schnittstellen* einen Energietransfer

vom lokalen Nano-Grid in das externe Energiesystem durchführt. Das nicht steuerbare Ereignis *calc* muss in jedem Zustand erlaubt werden und führt immer auf den Zustand *Hand* zurück.

Das **Schnittstellenmanagement für die Komponente AC/DC Öffentliches Netz** ist durch die Spezifikation K_P abgebildet und wird mittels folgenden Modell beschrieben.

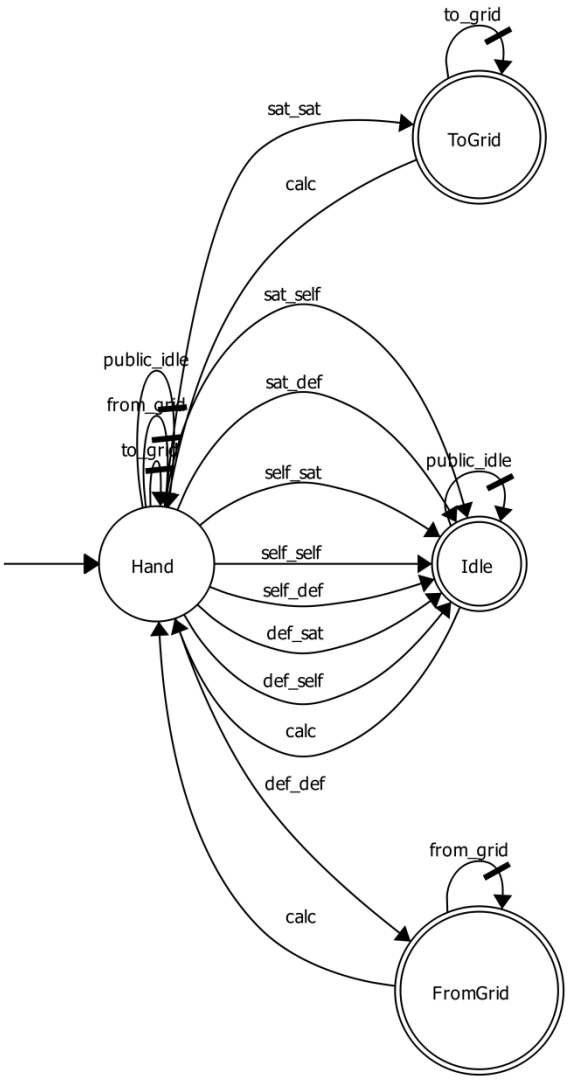

Abbildung 41: Generator der Spezifikation K_P für die Komponente *AC/DC Öffentliches Netz*

Im Initialzustand *Hand* werden alle steuerbaren Zustände der Strecke $G_{P,P}$ erlaubt. Sobald ein Statusereignis auftritt, wechselt der Generator in einen der Zustände *FromGrid*, ToGrid oder *Idle*. Es wird das jeweilige steuerbare Ereignis aus der Steuervorschrift aus Tabelle 10 erlaubt. Das Ereignis *calc* führt den Generator immer zum Zustand *Hand* zurück.

Für den Zustand *Hand* wir die Modellannahme getroffen, dass in der Programmsoftware die Generierung eines neuen Statusereignisses vor der Ansteuerung der Betriebszustände

stattfindet. Dadurch wird verhindert, dass trotz der erlaubten Ereignisse im Zustand *Hand*, die Betriebszustände nicht angesteuert werden, bevor ein neuer Gesamtzustand berechnet und als Ereignis ausgegeben wurde.

5.2.2 Spezifikation des Speichermanagements

Das Speichermanagement wird als Spezifikation K_{SOC} für die Komponente *DC/DC Speicher* modelliert. Sie schränkt das steuerbare Ereignisalphabet der Strecke $G_{S,P}$ in Abhängigkeit der nicht steuerbaren Ereignisse aus G_{SOC} ein. Auch für diese Spezifikation wird das Ereignis *error* aus der Strecke $G_{S,P}$ gestrichen.

Der modellierte Generator für die Spezifikation K_{SOC} ist folgender Abbildung dargestellt.

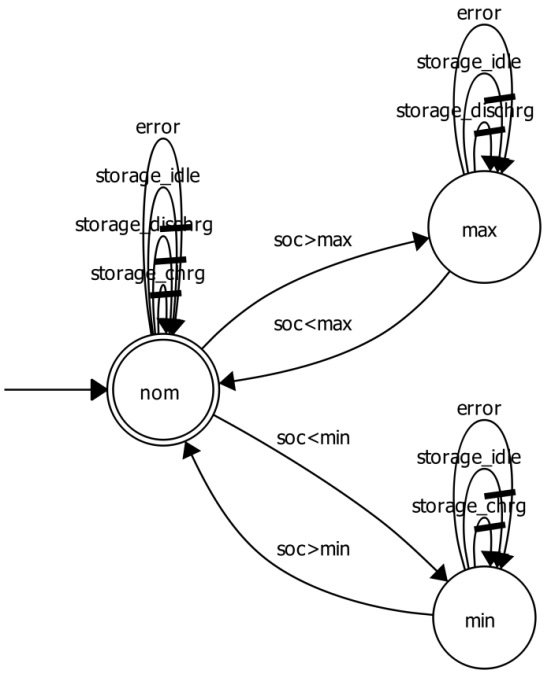

Abbildung 42: Generator der Spezifikation K_{SOC} für das Speichermanagement

Im Initialzustand *nom* werden alle steuerbaren Ereignisse erlaubt. Sobald einer der Grenzwert über- oder unterschritten wird und eines der Ereignisse *soc>max* oder *soc<min* auftritt, wechselt der Generator in einen der beiden Zustände *max* oder *min*. Der Zustand *max* signalisiert, dass der Speicher nicht weiter geladen werden darf. Daher wird das steuerbare Ereignis *storage_chrg* verhindert. Das gleiche Prinzip gilt für den Zustand *min*, in welchem das Ereignis *storage_dischrg* verhindert wird. Das Ereignis *storage_idle* wird immer erlaubt damit die Speicherkomponente immer in den Betriebszustand *Idle* geführt werden kann.

5.2.3 Spezifikation des Sicherheitsmanagements

Die Spezifikation K_{Safety} beschreibt das Sicherheitsmanagement, welches für die korrekte Ablaufreihenfolge der Zuschaltung der einzelnen DC/DC-Konverter nach Eintreten des globalen Ereignisses *error* zuständig ist. Sie schränkt alle aktiven Streckenkomponenten in ihren jeweiligen steuerbaren Ereignisalphabeten ein.

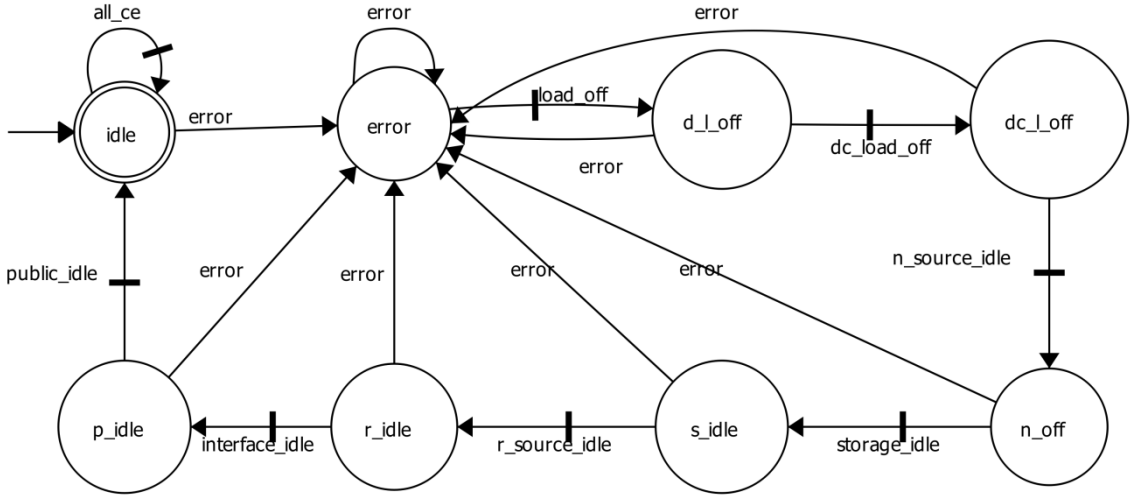

Abbildung 43: Generator der Spezifikation K_{Safety} für das Sicherheitsmanagement

Im Initialzustand sind alle Ereignisse der Steuerstrecke erlaubt. Dies wird vereinfach durch die Schlinge mit dem Ereignis *all_ce* dargestellt. Sobald das nicht steuerbare Ereignis *error* auftritt, findet ein Zustandswechsel statt. Ab hier werden sequentiell nur die Ereignisse aus der Ablaufreihenfolge aus Abbildung 17 erlaubt. Jedes der Ereignisse führt einen Zustandswechsel durch. Damit wird die Reihenfolge vorgegeben, in der die Ereignisse auftreten dürfen. Sobald die letzte Komponente *DC/DC Öffentliches Netz* ihr steuerbares Ereignis *public_idle* generiert, springt der Generator zurück auf den Initialzustand, welcher markiert ist und den erfolgreichen Ablauf der Sicherheitskette signalisiert.

5.3 Steuerungsentwurf

In diesem Kapitel werden die Supervisor aus den Modellen der Strecke und der Spezifikationen entworfen. Dabei werden zwei verschiedene Entwurfsansätze aus Kapitel 2.2.7 angewendet. Für das Sicherheitsmanagement wird der monolithische Ansatz gewählt wohingegen für das Leistungs- und Speichermanagement der lokal-modulare Ansatz verwendet wird. Die Entwurfsansätze leiten sich aus der Hardwarearchitektur des Nano-Grids ab.

Für die Steuerungssynthese wird das DESTool verwendet. Für jeden Steuerungsentwurf wird eine separate Datei erstellt, welche im Anhang auf CD gespeichert ist.

5.3.1 Monolithischer Steuerungsentwurf

Für den monolithischen Steuerungsentwurf werden zunächst die einzelnen aktiven ungesteuerten Streckenkomponenten mittels paralleler Komposition mit

$$G_{total} = G_R||G_N||G_S||G_L||G_{directL}||G_I||G_P$$

zum Gesamtstreckenmodell G_{total} zusammengeschaltet. Das Symbol || entspricht in dieser Gleichung dem SYPC-Operator. Die Spezifikation K_{Safety} greift auf G_{total} zu. Es wird anschließend die Steuerbarkeit von K_{Safety} bezüglich G_{total} mittels der im DESTool verfügbaren Funktion $IsControllable$ geprüft. Die Überprüfung ergibt folgendes Ergebnis:

$$IsControllable\left(K_{Safety}, G_{total}\right) \checkmark$$

Mittels der Funktion $SupConNB(K_{Safety}, G_{total})$ wird der Supervisor S_{Safety} berechnet. Die resultierende Steuerung S_{Safety} enthält insgesamt 1465 Zuständen und 20462 Transitionen. Eine Reduzierung der Steuerung mit $SupReduce(S_{Safety}, G_{total})$, lässt das DESTool abstürzen und kann dadurch nicht durchgeführt werden. Diese komplexe Zustands-Transitionsstruktur leitet sich aus dem unstrukturierten Gesamtmodell G_{total} ab. Der geschlossene Steuerkreis S_{Safety}/G_{total} ist in folgendem Blockschaltbild dargestellt.

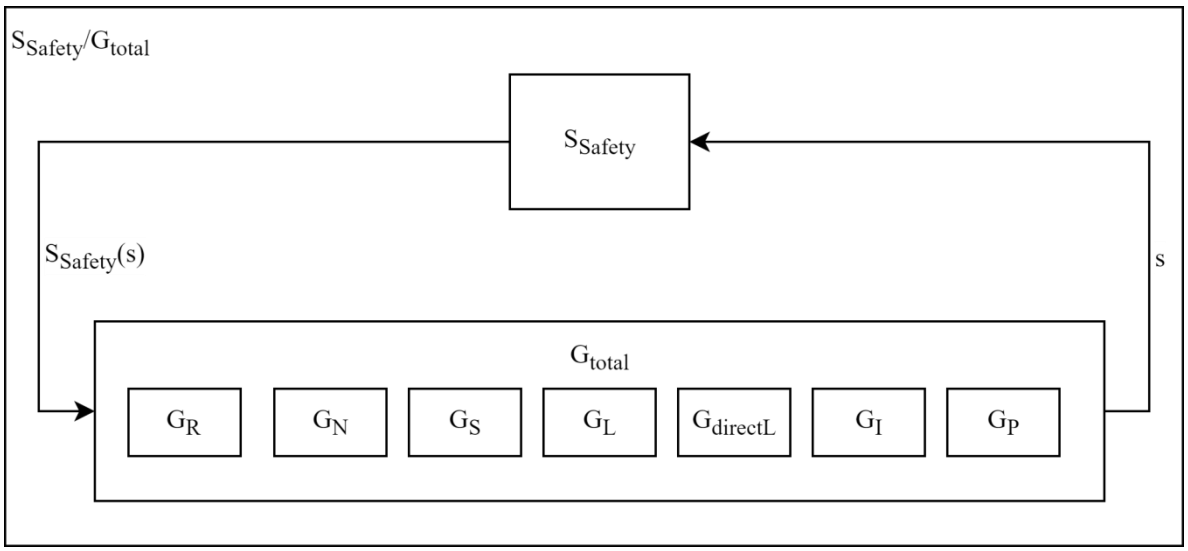

Abbildung 44: Blockschaltbild des monolithisch geschlossenen Steuerkreises S_{Safety}/G_{total}

Die Ereignisfolge s beinhaltet alle Ereignisse, welche durch die unstrukturierte Streckenkomponente G_{total} generiert werden. Die Ereignisfolge $S_{Safety}(s)$ beinhaltet die eingeschränkten steuerbaren sowie alle nicht steuerbaren Ereignisse aus s und gibt diese an G_{total} weiter.

5.3.2 Lokal-modularer Steuerungsentwurf

Für den lokal-modularen Entwurfsansatz werden die aktiven Streckenkomponenten ohne das Ereignis *error* betrachtet. Damit sind die aktiven Streckengeneratoren formal mit $G_{i,P} = \left(X_{i,P}, \Sigma_{i,P}, \delta_{i,P}, x_{i,P,0}, X_{i,P,m}\right)$ und $error \notin \Sigma_{i,P}$ mit $i = \{R, N, S, L, directL, I, P\}$ beschrieben. Mit dieser Modellanpassung sind alle aktiven und passiven Streckenkomponenten asynchron zueinander. In folgender Abbildung sind die Relationen zwischen den Ereignisalphabeten der Streckenkomponenten mit $G_i' = \left(X_i', \Sigma_i', \delta_i', x_{i,0}', X_{i,m}'\right)$ mit $i = \{'S,P','R,P','N,P','L,P','directL,P','P,P','I,P','Grid,V','SOC','Grid,Status'\}$ und den Spezifikationen $K_j \subseteq \Sigma_{Kj}$ mit $j = \{R, N, S, L, directL, I, P, SOC\}$ dargestellt.

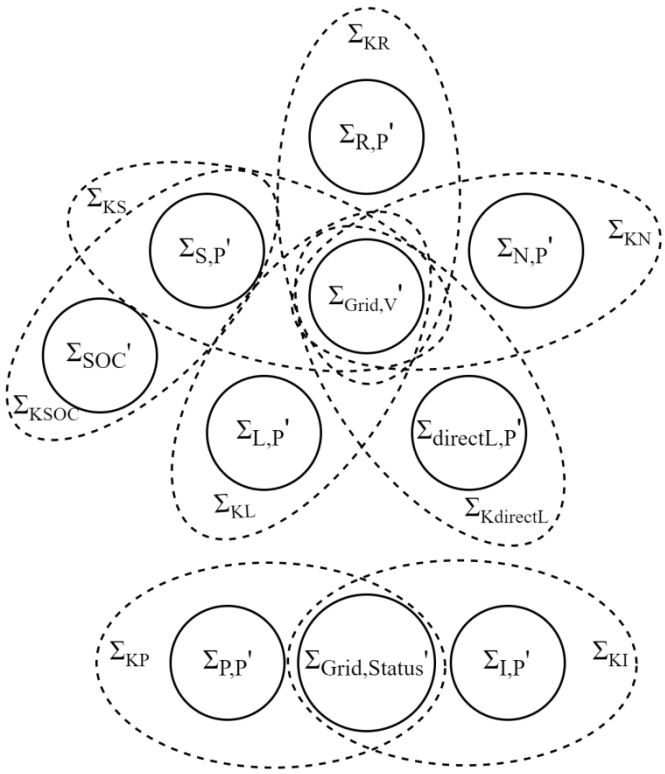

Abbildung 45: Relation der Ereignisalphabete für den lokal-modularen Steuerungsentwurf

Die ungesteuerte Strecke besteht insgesamt aus zehn Komponenten, welche über disjunkte Ereignisalphabete definiert sind und damit inhärent als feinstes Produktsystem mit $G_i = G_i'$, $i = \{'S,P',' R,P',' N,P',' L,P',' directL,P',' P,P',' I,P',' Grid,V',' SOC',' Grid,Status'\}$ vorliegen.

In diesem Entwurf werden die Spezifikation K_{SOC} und K_S zusammengefasst. Dafür werden die Spezifikationen mit den fehlenden Ereignissen der jeweils anderen erweitert. Für jedes fehlende Ereignis wird eine Schlinge an jeden Zustand angefügt. Im DESTool wird dafür der Befehl $SelfLoop(K_{SOC}, (\Sigma_S - \Sigma_{SOC}))$ und $SelfLoop(K_S, (\Sigma_{SOC} - \Sigma_S))$ verwendet. Anschließend werden beide Spezifikationen mit der Produktkomposition SPC zur Gesamtspezifikation K_{SOCXS} mit

$$K_{SOCXS} = K_{SOC} ||_{SPC} K_S$$

zusammengeführt. Es ergeben sich die lokalen Systeme G_{Xi} mit $i = \{R, N, S, L, directL, I, P\}$ durch parallele Komposition der Streckenkomponenten, welche durch die einzelnen Spezifikationen K_j mit $j = \{R, N, SOCXS, L, directL, I, P\}$ zwanghaft synchronisiert werden, wie folgt:

$$G_{XR} = G_R||_{SYPC}G_{Grid,V},$$

$$G_{XN} = G_N||_{SYPC}G_{Grid,V},$$

$$G_{XS} = G_S||_{SYPC}G_{Grid\_V}||_{SYPC}G_{SOC},$$

$$G_{XL} = G_L||_{SYPC}G_{Grid,V},$$

$$G_{XdirectL} = G_{directL}||_{SYPC}G_{Grid,V},$$

$$G_{XI} = G_I||_{SYPC}G_{Grid,Status},$$

$$G_{XP} = G_P||_{SYPC}G_{Grid,Status}.$$

Mittels der lokalen Systeme werden nun die angepassten Spezifikationen E_{Xi} mit $i = \{R, N, S, L, directL, I, P\}$ unter Verwendung der Gleichung (2.54) wie folgt berechnet:

$$E_{XR} = K_R||_{SYPC}G_{XR},$$

$$E_{XN} = K_N||_{SYPC}G_{XN},$$

$$E_{XS} = K_{SOCXS}||_{SYPC}G_{XS},$$

$$E_{XL} = K_L||_{SYPC}G_{XL},$$

$$E_{XdirectL} = K_{directL}||_{SYPC}G_{XdirectL},$$

$$E_{XI} = K_I||_{SYPC}G_{XI},$$

$$E_{XP} = K_P||_{SYPC}G_{XP}.$$

Die angepassten Spezifikationen werden im nächsten Schritt auf Steuerbarkeit bezüglich der lokalen Systeme geprüft. Die Überprüfung ergibt folgende Ergebnisse:

$$IsControllable(E_{Xi}, G_{Xi})\ \checkmark \text{ mit } i = \{R, N, S, L, directL, I, P\}.$$

Die angepassten Spezifikation sind alle steuerbar bezüglich ihrer lokalen Systeme, wodurch keine supremale steuerbare Teilsprache $E_{Xi}^{\uparrow C}$ der einzelnen Spezifikationen benötigt wird. Der nächste Analyseschritt ist die Prüfung auf lokale Modularität, welche formal nach Gleichung (2.57) mit

$$||_{i\in l}\overline{E_{Xi}} = \overline{||_{i\in l}E_{Xi}} \text{ für } l = \{R, N, S, L, directL, I, P\}$$

erfolgt. Im DESTool wird für diese Prüfung der Befehl $isNonBlocking$ sukzessive für zwei Spezifikationen angewendet. Das Ergebnis der Überprüfung wird zusammengefasst dargestellt als:

$$isNonBlocking(E_{XR}, E_{XN}, E_{XS}, E_{XL}, E_{XdirectL}, E_{XI}, E_{XP},)\ \checkmark.$$

Damit sind alle Spezifikationen nichtblockierend und können zur Berechnung der lokalen Supervisor S_i mit $i = \{R, N, S, L, directL, I, P\}$ genutzt werden. Die Berechnung der lokalen Supervisor erfolgt analog zum monolithischen Entwurf mittels der Funktion $SupConNB(E_{Xi}, G_{Xi})$. In folgender Übersicht sind die Ergebnisse der Supervisor mit ihren jeweiligen Zuständen (n) und Transitionen (t) dargestellt.

Tabelle 23: Übersicht der Anzahl an Zuständen und Transitionen der lokal-modularen Steuerungen

Supervisor	**Berechnung**	**n**	**t**
S_R	$SupConNB(E_{XR}, G_{XR})$	21	54
S_N	$SupConNB(E_{XN}, G_{XN})$	21	54
S_S	$SupConNB(E_{XS}, G_{XS})$	63	266
S_L	$SupConNB(E_{XL}, G_{XL})$	21	66
$S_{directL}$	$SupConNB(E_{XdirectL}, G_{XdirectL})$	14	37
S_I	$SupConNB(E_{XI}, G_{XI})$	30	78
S_P	$SupConNB(E_{XP}, G_{XP})$	30	78
	Summe	**200**	**633**

In folgender Abbildung ist der geschlossene Steuerkreis S_{mod}/G mit seinen einzelnen Supervisor und den lokalen Systemen als Blockschaltbild dargestellt.

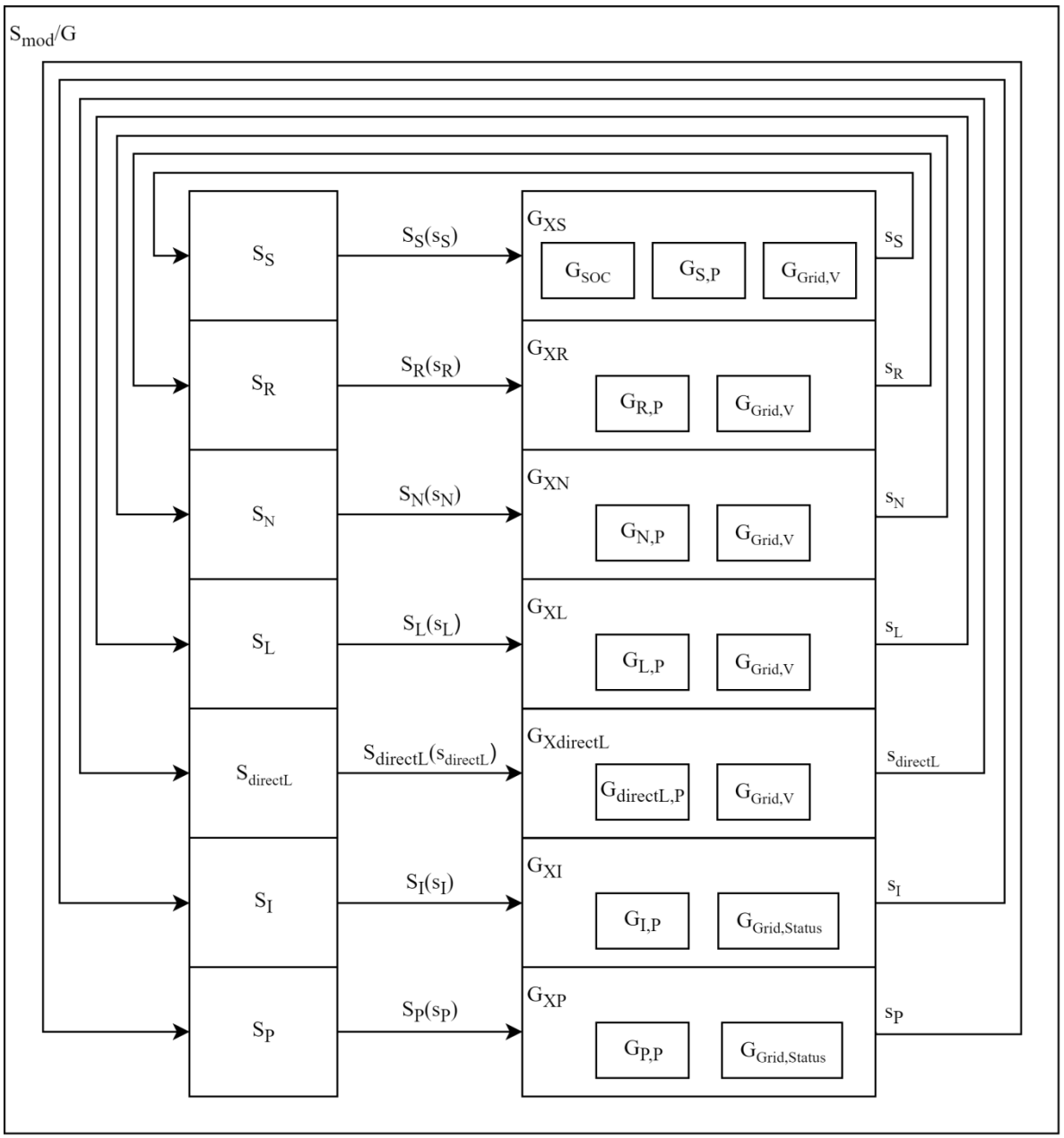

Abbildung 46: Blockschaltbild des lokal-moadular geschlossenen Gesamtsteuerkreises S_{mod}/G

Jedes lokale System generiert einen Ereignisstring s_i mit $i = \{R, N, S, L, directL, I, P\}$, welcher durch die einzelnen Supervisor eingeschränkt wird. Daraus ergibt sich das gesteuerte Verhalten für jedes lokale geschlossene System, sowie für das Gesamtsystem.

Zur Eliminierung von redundanten Zuständen werden abschließend die einzelnen Supervisor mit der Funktion $SupReduce(S_i, G_{Xi})$ mit $i = \{R, N, S, L, directL, I, P\}$ reduziert. Folgende Übersicht zeigt das Ergebnis der Reduktion anhand der Anzahl an Zuständen und Transitionen der neuen berechneten Supervisor $S_{i,red}$ mit $i = \{R, N, S, L, directL, I, P\}$.

Tabelle 24: Übersicht der Anzahl an Zuständen und Transitionen nach der reduizierten lokal-modularen Steuerungen

Supervisor	**Berechnung**	**n**	**t**
$S_{R,red}$	$SupReduce(S_R, G_{XR})$	3	7
$S_{N,red}$	$SupReduce(S_N, G_{XN})$	4	9
$S_{S,red}$	$SupReduce(S_S, G_{XS})$	8	11
$S_{L,red}$	$SupReduce(S_L, G_{XL})$	3	7
$S_{directL,red}$	$SupReduce(S_{directL}, G_{XdirectL})$	2	3
$S_{I,red}$	$SupReduce(S_I, G_{XI})$	4	13
$S_{P,red}$	$SupReduce(S_P, G_{XP})$	4	13
	Summe	**28**	**63**

Es ist zu erkennen, dass die Zustands-Transitionsstruktur eine deutlich reduziertere Komplexität vorweist. Es werden insgesamt 86% weniger Zustände und 90% weniger Transitionen benötigt.

6.Simulation und Ergebnisbewertung

In diesem Kapitel werden die mittels der Simulationsumgebung im DESTool synthetisierten Supervisor aus Kapitel 5.3 verifiziert. Dazu wird die Durchführung der einzelnen Simulationsschritte beschrieben und anschließend die Ergebnisse der Simulation bewertet.

6.1 Simulationsdurchführung

Für die Simulation werden jeweils die geschlossenen Steuerkreise mit der monolithischen Steuerung und den lokal-modular entworfenen Steuerungen betrachtet.

6.1.1 Verifizierung der monolithischen Steuerung

Für die **monolithische Steuerung** wird der geschlossene Steuerkreis mittels der Produktkomposition aus dem Supervisor S_{Safety} und der Gesamtstrecke G_{total} mit $S_{Safety}/G_{total} = S_{Safety} ||_{SPC} G_{total}$ erstellt. Im DESTool wird das erstellte System über das Kontextmenü als *Plant* deklariert. Zur Überprüfung der Korrektheit wird das nicht steuerbare Ereignis *error* händisch angesteuert, da die gewünschte Einschränkung der Ablaufreihenfolge abhängig vom Ereignis *error* gesteuert werden soll.

In der folgenden Abbildung ist die Animationsumgebung im DESTool zu erkennen. Im oberen Bereich ist der geschlossene Steuerkreis mit seinen komponierten Einzelzuständen, welche sich zu Beginn alle in *Idle* bzw. *Off* befinden, abgebildet. Im unteren Abschnitt sind die erlaubten (linke Spalte) und nichterlaubten (rechte Spalte) Ereignisse aufgelistet. Zur Übersichtlichkeit wurden die Bereiche separat als Bildausschnitt herausgeschnitten und zusammengefügt. Die erlaubten Ereignisse sind nicht vollständig erkennbar. Es wird auf den Anhang verwiesen, in welchem die DESTool Datei hinterlegt ist und damit auch die Animationsumgebung vollständig eingesehen werden kann.

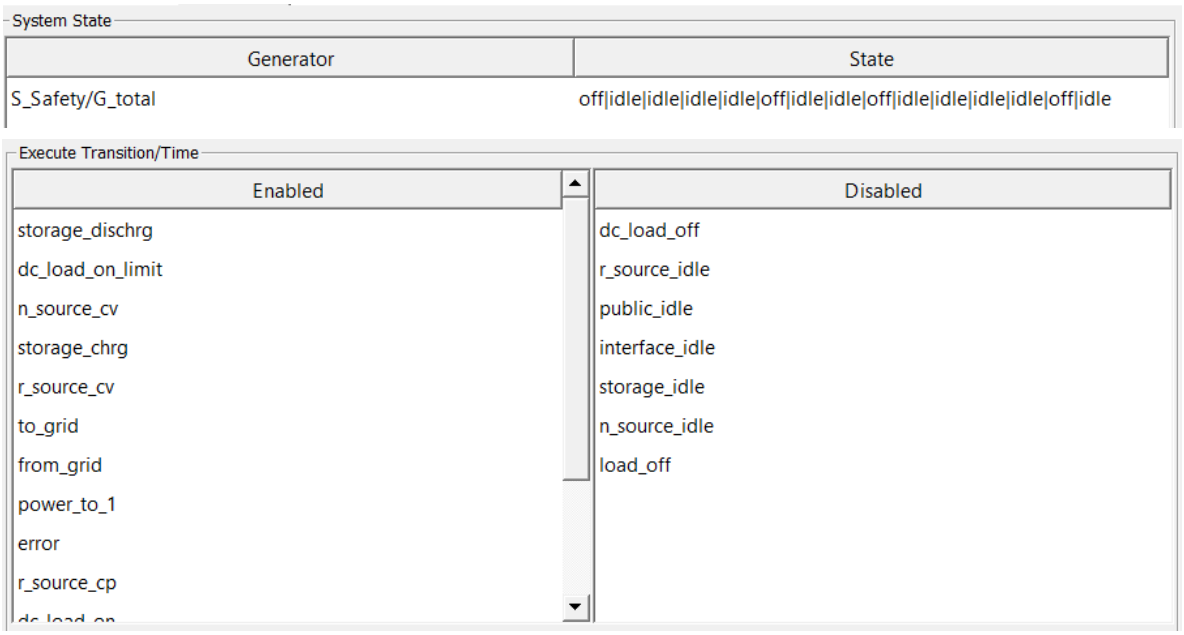

Abbildung 47: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_{Safety}/G_{total} im Initialzustand

Es ist zu erkennen, dass alle Ereignisse, die das Gesamtsystem generieren kann, erlaubt werden und keine Einschränkung durch die Steuerung stattfinden. Erst nach Ansteuerung des Ereignisses *error*, welches in der realen Anlage das Betätigen eines Sicherheitsschalters darstellt, wird das erlaubte Ereignisalphabet, wie in folgender Abbildung dargestellt, eingeschränkt. Es wird nur noch der untere Teil der Animationsumgebung abgebildet.

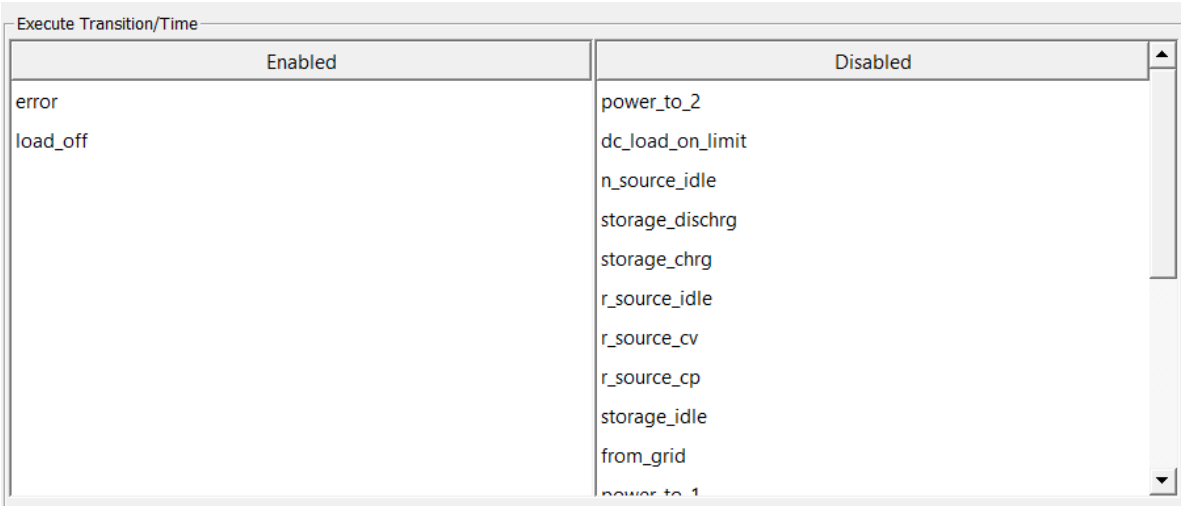

Abbildung 48: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_{Safety}/G_{total} nach Eintreten des Ereignisses *error*

Es ist nur das steuerbare Ereignis *load_off* sowie das nicht steuerbare Ereignis *error* zugelassen. Alle weiteren steuerbaren Ereignisse werden nicht zugelassen und sind auf der

rechten Seite aufgelistet. Im nächsten Schritt wird das Ereignis *load_off* aktiviert. Das Ergebnis wird in folgender Abbildung dargestellt.

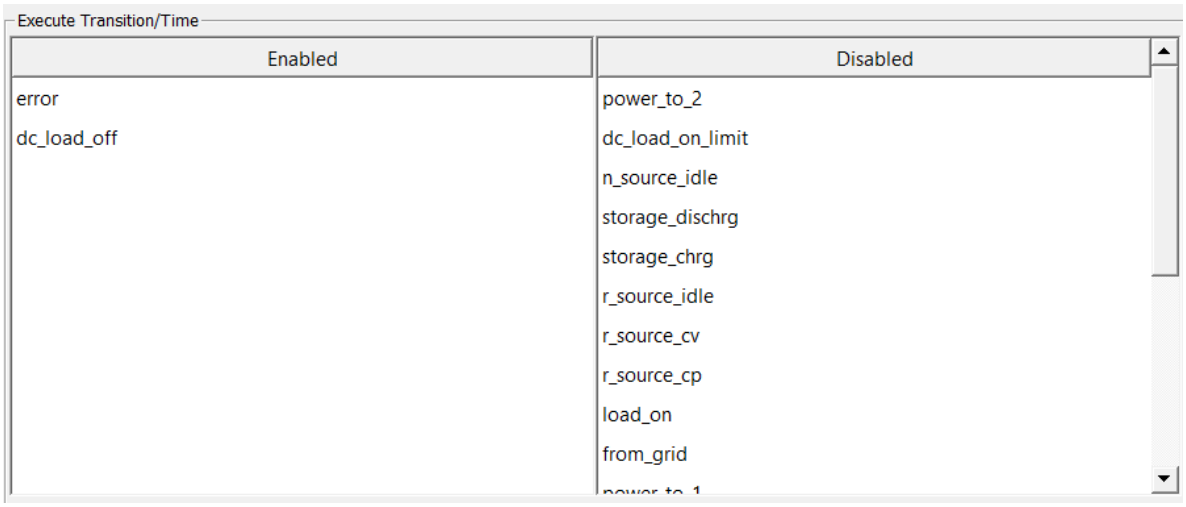

Abbildung 49: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_{Safety}/G_{total} nach Eintreten der Ereigniss *error* und *load_off*

Es wird nur noch das steuerbare Ereignis *dc_load_off* erlaubt. Das Ereignis *error* wird in allen weiteren Ereignisschritten immer erlaubt. Die Reihenfolge der ersten beiden Schritte ist damit erfüllt. Auf eine weitere Abbildung der erlaubten Ereignisse wird verzichtet und erneut auf das Programm im Anhang verwiesen, in dem alle Ereignisschritte sukzessive durchgeführt werden können.

6.1.2 Verifizierung der lokal-modularen Steuerungen

Für die Verifikation des lokal-modularen Steuerungsentwurfes werden die lokalen Systeme mit den jeweiligen Supervisor mittels Produktkomposition zu lokalen geschlossenen Steuerkreisen zu $S_i/G_{Xi} = S_i \,||_{SPC} G_{Xi}$ mit $i = \{R, N, S, L, directL, I, P\}$ zusammengeführt. Die geschlossenen Steuerkreisen werden im DESTool jeweils als *Plant* deklariert und anschließend in der Animationsumgebung abgebildet. Die Spezifikationen für das Erzeugermanagement, welches mit dem Speichermanagement zusammengeführt ist, sowie für das Lastmanagement werden separat zum Schnittstellenmanagement verifiziert, da dieses komplett nebenläufig abläuft. Dies ist aus den Relationen der Ereignismengen ersichtlich (siehe Abbildung 45).

Für die **Verfikation des Erzeuger- und Lastmanagements** werden die nicht steuerbaren Ereignisse *<Vi* und *>Vi* mit $i = [0..5]$ sowie *soc<min*, *soc>min*, *soc>max* und *soc<max* der lokalen Streckensysteme in der Animationsumgebung händisch angesteuert. Diese repräsentieren im realen Nano-Grid die Schwellwertereignissen der Verteilnetznetzspannung

sowie des SOC. In der folgenden Abbildung ist die Animationsumgebung mit den lokalen geschlossenen Steuerkreisen S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ im Initialzustand dargestellt. Es werden alle erlaubten und nicht erlaubten Ereignisse der Steuerkreise zusammen aufgeführt.

Abbildung 50: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ im Initialzustand

Im Initialzustand findet keine Einschränkung statt. Damit befindet sich das Nano-Grid, wie in den Spezifikationen modelliert, im Handbetrieb, in welchem alle steuerbaren Ereignisse erlaubt sind. Im nächsten Schritt soll das Unterschreiten des ersten Schwellwertes durch das Ansteuern des Ereignisses $<V0$ simuliert werden. Die folgende Abbildung stellt die erlaubte Ereignismenge dar. Auf die Darstellung der betrachteten Steuerkreise wird verzichtet. Im weiteren Verlauf bleiben die aktiven Streckenkomponenten im Initialzustand. Es werden daher immer die erlaubten Ereignisse vom Initialzustand aus dargestellt.

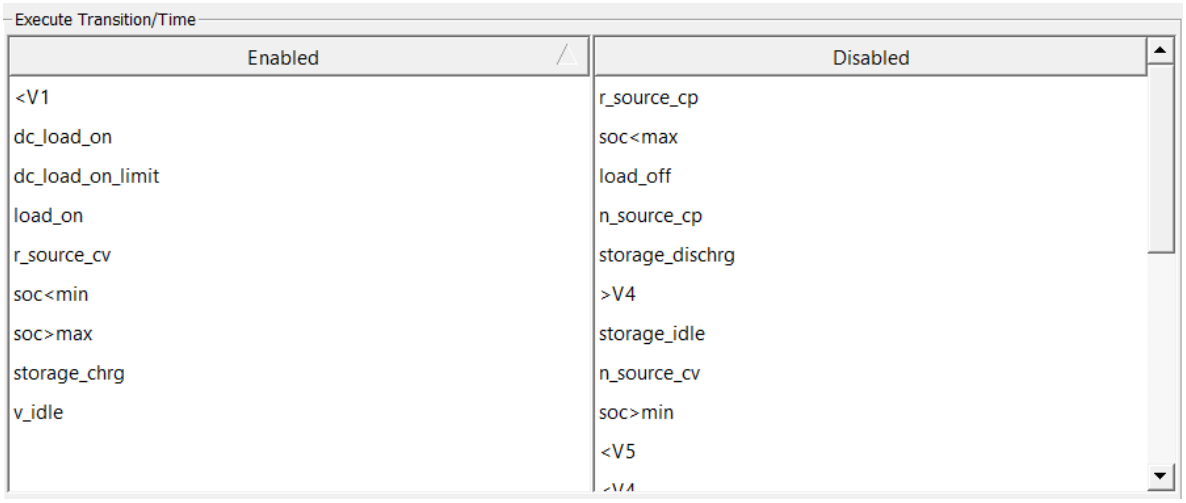

Abbildung 51: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ nach Eintreten des Ereignisses *<V0*

Es ist zu erkennen, dass die Erzeugerkomponenten in ihren steuerbaren Ereignissen eingeschränkt werden. So kann die im Steuerkreis durch Komposition erhaltene Streckenkomponente $G_{R,P}$, welche die Betriebszustände der erneuerbaren Komponente *DC/DC Regenerativ* beschreibt, nicht das Ereignis *r_source_cp* ausführen und somit nicht den Betriebszustand *Constant Power* ansteuern. Die Speicherkomponente G_S darf sich nicht entladen, was durch das Verhindern des Ereignisses *storage_dischrg* erfolgt. Im nächsten Schritt wird das Ereignis *soc>max* angesteuert, um den maximalen Ladezustand des Speichers zu simulieren. Die eingeschränkten Ereignisalphabete sind in der folgenden Abbildung ersichtlich.

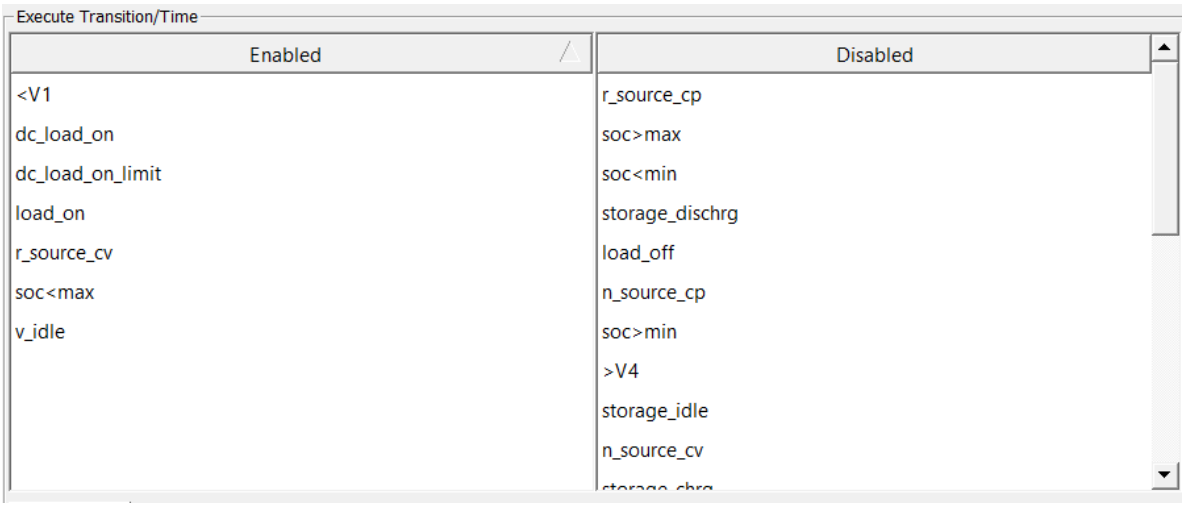

Abbildung 52: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ nach Eintreten des Ereignisse *<V0* und *soc>max*

Das Ereignis *storage_chrg* wird nicht mehr erlaubt und verhindert somit das Überladen der Speicherkomponente. Damit wird sowohl die Spezifikation des Speicher- als auch des Erzeugermanagements erfüllt. In der weiteren Simulationsdurchführung wird das Überschreiten der einzelnen Spannungsschwellwerte durch das Ansteuern der entsprechenden Ereignisse simuliert und die erlaubte Ereignismenge geprüft. Es wird auf die Darstellung der gesamten Überprüfung verzichtet und auf die DESTool Projektdatei „Modell_nanogrid_lokal-modular.pro“ im Anhang verwiesen.

Zur **Verifikation des Lastmanagements** wird das Auftreten des Schwellwertes V_3 durch Ansteuern des Ereignisses <V3 simuliert. Die verfügbare erlaubte Ereignismenge ist in der folgenden Abbildung abgebildet.

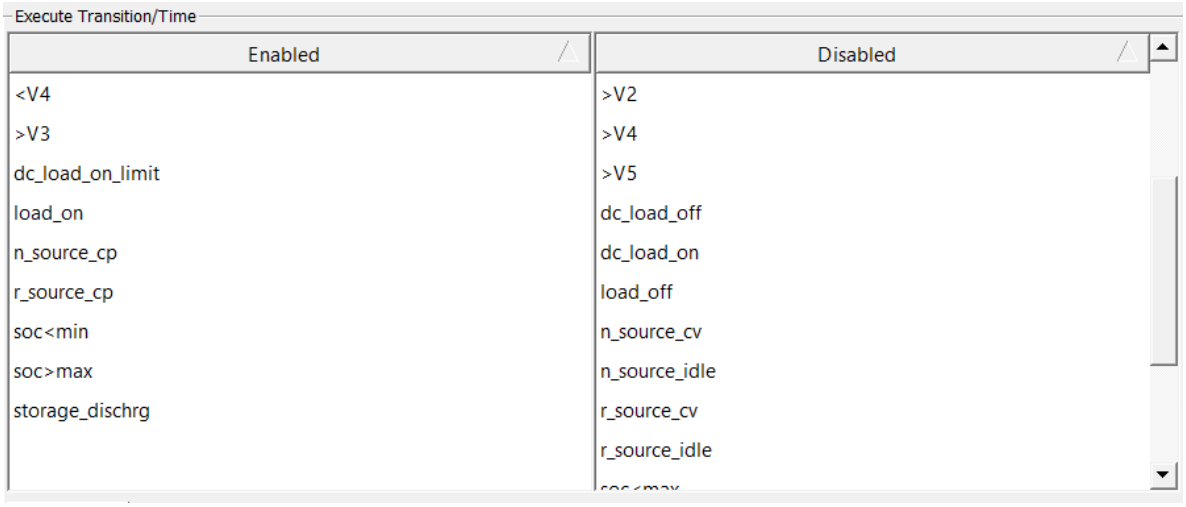

Abbildung 53: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ nach Eintreten des Ereignisse *<V0,<V1,<V2* und *<V3*

Durch die Unterschreitung des Schwellwertes V_3 darf die Lastkomponente G_L nicht das Ereignis *dc_load_on* generieren können. Die Lastkomponente $G_{directL}$ hingegen ist nicht beschränkt. Erst nach Ansteuern des Schwellwertereignisses *<V5* wird auch das Ereignis *load_on* der Komponente $G_{directL}$ verhindert, wie in der folgenden Abbildung zu erkennen ist.

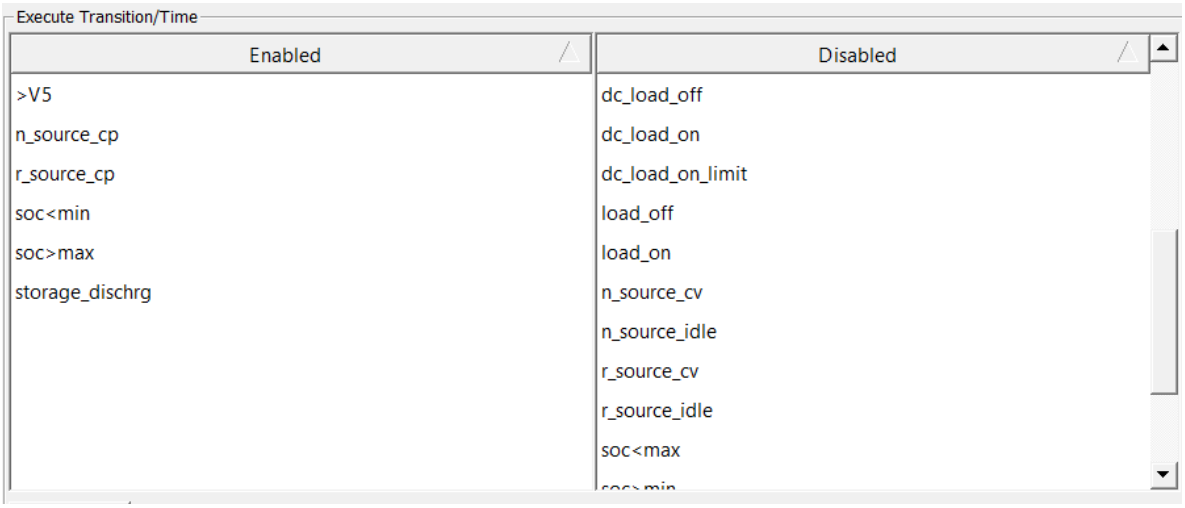

Abbildung 54: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_i/G_{Xi} mit $i = \{R, N, S, L, directL\}$ nach Eintreten des Ereignisse *<V0,<V1,<V2, <V3,<V4* und *<V5*

Damit werden die steuerbaren Ereignisse in Abhängigkeit der nicht steuerbaren Ereignisse nach der geforderten Spezifikation verhindert.

Abschließend wird die die Verifikation des **Schnittstellenmanagements** durchgeführt. Dafür werden die ungesteuerten Ereignisse der lokal geschlossenen Steuerkreise S_I/G_{XI} und S_P/G_{XP} angesteuert. Die Durchführung erfolgt analog zu den für das Erzeuger- und Lastmanagement beschriebenen Simulationsschritten indem die nicht steuerbaren Ereignisse aus dem lokal vorhanden Streckengenerator $G_{Grid,Status}$ angesteuert werden.

Im Initialschritt werden alle gesteuerten Ereignisse erlaubt. Nachfolgend wird nur die Ansteuerung des Ereignisses *self_def* durchgeführt und anschließend die erlaubte Ereignismenge dargestellt. Für eine vollständige Überprüfung wird auf die DESTool Datei im Anhang verwiesen.

Abbildung 55: Übersicht der erlaubten und nicht erlaubten Ereignisse von S_I/G_{XI} und S_P/G_{XP} nach Eintreten des Ereignisses *self_def*

Die Schnittstellenkomponente G_I kann nur noch das steuerbare Ereignis *power_ext* genierenen, was der lokal angepassten Spezifikation E_{XI} entspricht. Die Komponente G_P befindet sich bereits im Zustand *Idle*. Ein Zustandswechsel zu einem der weiteren Betriebszustände wird auch hier verhindert.

6.2 Ergebnisbewertung

Durch die im vorherigen Kapitel durchgeführten Simulationsschritte sind die entworfenen Steuerungen verifiziert worden. Die Streckenkomponenten wurden durch geeignete Modellannahmen modularisiert und strukturiert. Die entsprechenden Spezifikationen wurden so modelliert, dass die informell gestellten Steuerungsvorschriften aus Kapitel 3.3 korrekt ablaufen. Damit wird die geforderte Steuerungsaufgabe erfüllt.

Die Entwurfsansätze wurden unter Berücksichtigung der gegebenen Systemarchitektur so gewählt, dass eine möglichst verteilte Implementierung stattfinden kann. Die entworfenen lokal-modularen Steuerungen enthalten damit eine geringe algorithmische Komplexität und können somit gut in eingebettete Rechnersysteme implementiert werden. Eine Ausnahme stellt die Steuerung für das Sicherheitsmanagement dar.

Durch die Unterteilung in aktive und passive Streckenkomponenten, reagiert die Steuerung auf Änderungen der Verteilnetzspannung und dem SOC. Zusätzlich wird die Kopplung von

mindestens zwei Nano-Grid-Systemen gesteuert, welches zu einer einfachen Skalierung der Systemgröße führen kann und damit zu einem modularen Aufbau eines wachsenden dezentralen Energiesystems. Durch die Ansteuerung des Leistungsaustausches mit dem öffentlichen Netz wird die Versorgungssicherheit in jedem Fall gewährleistet, sofern das öffentliche Netz eine stabile Versorgung bereitstellen kann.

Die Anwendbarkeit ereignisdiskreter Entwurfsverfahren ist speziell durch die kompositionale Modellbildung gut geeignet. Gerade bei einer steigenden Anzahl von gekoppelten Erzeuger-, Speicher- oder Lastkomponenten und einer hierarchischen Systemtopologie, kann eine intuitive Implementierung unübersichtlich und fehleranfällig werden. Allerdings müssten dazu konkrete Unterscheidungskriterien getroffen und verglichen werden.

7.Zusammenfassung und Ausblick

Abschließend wird in diesem Kapitel eine Zusammenfassung durchgeführt sowie ein Ausblick zu weiterführenden Arbeiten gegeben.

7.1 Zusammenfassung

Im Rahmen dieser Masterarbeit wurden die Methoden der Supervisory Control Theory (SCT) für die Steuerung eines dezentralen Energiesystems in Form eines Nano-Grids angewendet. Dazu wurde ein entsprechendes Nano-Grid-Modell unter Einbezug der Ergebnisse aktueller Forschungsprojekte konzipiert.

Die Modellbildung des konzipierten Nano-Grids erfolgte durch Modularisierung in einzelne Komponenten und die strukturierte Kopplung durch definierte Kompositionsoperatoren. Dabei fand eine Abstraktion auf logischer Ebene statt. Es wurde eine formale Beschreibung der Steuerstrecke in Generatorform durchgeführt.

Die geforderten Steuerungsaufgaben wurden informell abgebildet und anschließend formal als Spezifikationen in Generatorform beschrieben. Durch Anwendung der beschriebenen Analysemethoden wurde das System auf Steuerbarkeit und Blockierung hin untersucht.

Der Steuerungsentwurf erfolgte durch Anwendung des monolithischen und lokal-modularen Ansatzes. Die entworfenen Steuerungen wurden abschließend durch Simulation der geschlossenen Steuerkreise verifiziert und als korrekt bewertet.

7.2 Ausblick

Als weiterführende Arbeit ist die Implementierung der entworfenen Steuerung auf einer realen Systemumgebung zu nennen. Hierfür ist eine unterlagerte Steuerung zu realisieren, welche die steuerbaren Ereignisse aus den entworfenen Steuerungen in Ansteuersignale für die Umschaltung der Betriebszustände durchführt. Weiterhin ist ein Ereignisgenerator zu implementieren, welcher die ungesteuerten Ereignisse der modellierten Streckenkomponenten generiert.

Des Weiteren kann die Auslegung der vorgestellten Voltage-Droop-Regelung sowie der MPPT-Regelung als unterlagerte Regelstrategie durchgeführt werden. Dazu sollen auch Möglichkeiten hinsichtlich dezentraler Regelung genannt werden. In [37] wird dazu ein Konzept vorgestellt.

Zudem bietet die überlagerte Steuerebene aus Abbildung 14 weiteres Entwicklungspotential hinsichtlich prognosebasierter Verfahren an. Dafür ist eine Betrachtung von ereignisdiskreten Modellmethoden auf Basis der in dieser Arbeit entwickelten Modelle möglich.

Ein weiteres interessantes Themenfeld bietet die detailliertere Betrachtung der Kopplungsmöglichkeiten einzelner Nano-Grids. Dazu können verschiedene Steuerstrategie ebenfalls aufbauend auf der in dieser Arbeit erstellten Modellbetrachtung durchgeführt werden. In [38] wird ein selbstorganisierendes Steuerkonzept vorgestellt. Daher soll auf Untersuchung ereignisdiskreter Modellverfahren zur Ansteuerung solcher Kopplungsstrategien erwähnt werden.

Literaturverzeichnis

[1] J. Zheming, Z. Jin, H. A. Khan, N. A. Zaffar, J. C. Vasquez und J. M. Guerrero, „A Decentralized Control Architecture applied to DC Nanogrid Clusters for Rural Electrification in Developing Regions," *IEEE Transactions on Power Electronics,* 19 April 2018.

[2] B. Nordman und K. Christensen, „Local Power Distribution with Nanogrids," *2013 International Green Computing Conference Proceedings,* 29 Juni 2013.

[3] S. Bowjiya, S. Hema, C. Suresh Kumar und R. Kalaivani, „Design and Implementation of Smart Nanogrid," *International Journal of Pure and Applied Mathematics, Volume 118, No. 24,* Mai 2018.

[4] J. Lunze, Regelungstechnik 1 - Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen, Berlin: Springer Verlag, 2010.

[5] F. Wenck, *Masterstudiengang Automatisierung, Ereignisdiskrete System - Einführung und Terminologie,* Hamburg: HAW Hamburg, 2016.

[6] J. Lunze, Ereignisdiskrete Systeme - Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen, München: Oldenbourg Wisschenschaftsverlag, 2012.

[7] F. Wenck, Modellbildung, Analyse und Steuerungsentwurf für gekoppelte ereignisdiskrete Systeme, Aachen: Shaker Verlag, 2006.

[8] C. G. Cassandras und S. Lafortune, Introduction to Discrete Event Systems, Kluwer: Springer Science+Business Media, 2008.

[9] W. Wonham, „Supervisory Control of Discrete-Event Systems," Dep. of Electrical and Computer Engineering, University of Toronto, Toronto, 2004.

[10] W. M. Wonham und P. J. Ramadge, „On the supremal controllable sublanguage of a given language," in *The 23rd IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, 1984.

[11] P. Ramadge und W. Wonham, „Supervisory Control of a Class of Discrete Event," *SIAM J. Control and Optimization 25,* pp. 206-230, Januar 1987.

[12] J. Moody und P. Antsaklis, Supervisory Control of Discrete Event Systems Using Petri Nets, Kluwer: Springer US, 1998.

[13] U. Kiencke und F. Puente León, Ereignisdiskrete Systeme: Modellierung und Steuerung verteilter Systeme, München: Oldenbourg Verlag, 2013.

[14] P. J. Ramadge, Supervision of discrete event processes, Toronto: Dep. of Electrical and Computer Engineering, University of Toronto, Dissertation, 1983.

[15] P. Varaiya und A. B. Kurzhanski, Discrete Event Systems: Models and Applications; Proceedings of an IIASA Conference, Sopron, Hungary, August 3-7, 1987, Heidelberg: Springer Verlag, 1988.

[16] P. J. Ramadge und W. M. Wonham, „The Control of Discrete Event Systems," *Proceedings of the IEEE, Volume: 77 , Issue: 1*, pp. 81 - 98, Januar 1989.

[17] W. M. Wonham, „Posted Research Material on Supervisory Control of Discrete-Event Systems: Design Software: TCT," University of Toronto, Department of Electrical and Computer Engineering, 01 05 2017. [Online]. Available: https://www.control.utoronto.ca/DES/Research.html. [Zugriff am 3 06 2018].

[18] R. Cieslak, C. Desclaux, A. Fawaz und P. Varaiya, „Supervisory control of discrete-event processes with partial observations," *IEEE Transactions on Automatic Control, Volume: 33 , Issue: 3,* März 1988.

[19] F. Lin und W. M. Wonham, „Decentralized control and coordination of discrete-event systems with partial observation," *EEE Transactions on Automatic Control 35,* pp. 1330-1337, Dezember 1990.

[20] W. M. Wonham und P. J. Ramadge, „Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals and Systems, Volume 1, Issue 1,* pp. 13-30, Februar 1988.

[21] M. H. d. Queiroz und J. E. R. Cury, „Modular Supervisory Control of Large Scale Discrete Event Systems," *Discrete Event Systems ,* pp. 103-110, November 2000.

[22] M. H. d. Queiroz und J. E. R. Cury, „Modular Control of Composed Ssystems," *Proceedings of the American Control Conference,* pp. 4051-4055, Juni 2000.

[23] R. Su und W. Wonham, „Supervisor Reduction for Discrete-Event Systems," *Discrete Event Dynamic Systems, Volume 14, Issue 1,* pp. 31-53, Januar 2004.

[24] L. F. und W. Wonham, „Decentralized supervisory control of discrete-event systems," *Information Sciences, Volume 44, Issue 3,* pp. 199-224, April 1988.

[25] F. Lin und W. Wonham, „On observability of discrete-event systems," *Information Sciences, Volume 44, Issue 3,* pp. 173-198, April 1988.

[26] F. S. T. J., v. d. M.-F. J. M., S. R. und R. J. E., „Application of supervisory control theory to theme park vehicles," *Discrete Event Dynamic Systems,* pp. 511-540, Dezember 2012.

[27] G. Brauner, Energiesysteme: regenerativ und dezentral - Strategien für die Energiewende, Wiesbaden: Springer Vieweg, 2016.

[28] J. Specovius, Grundkurs Leistungselektronik - Bauelemente, Schaltungen und Systeme, Wiesbaden: Springer Vieweg, 2018.

[29] D. Burmester, R. Rayudu, W. Seah und D. Akinyele, „A review of nanogrid topologies and technologies," *Renewable & Sustainable Energy Reviews,* pp. 760-775, 2017.

[30] T. Moor, „DESTool: About," Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, 2008-2016. [Online]. Available: https://www.rt.tf.fau.de/FGdes/destool/index.html. [Zugriff am April 2018].

[31] K. Sun, L. Zhang, Y. Xing und J. M. Guerrero, „A Distributed Control Strategy Based on DC Bus Signaling for Modular Photovoltaic Generation Systems With Battery Energy Storage," *IEEE Transactions on Power Electronics,* 2011.

[32] J. Schönberger, R. Duke und S. D. Round, „Decentralised source scheduling in a model nanogrid using DC bus signalling," *Australian Journal of Electrical and Electronics Engineering,* Januar 2005.

[33] J. Schönberger, R. Duke und S. D. Round, „DC-Bus Signaling: A Distributed Control Strategy for a Hybrid Renewable Nanogrid," *IEEE Transactions on Industrial Electronics, vol. 53,* pp. 1453-1460, Oktober 2006.

[34] T. L. Nguyen und G. Griepentrog, „A self-sustained and flexible decentralized control strategy for DC nanogrids in remote areas/islands," *2017 IEEE Southern Power Electronics Conference (SPEC),* pp. 1-6, 2017.

[35] N. H. Saad, A. A. El-Sattar und N. M. Rady, „Calculation of voltage thresholds for source schedulaing in a hybrid renweable nanogrid," *Journal of Electrical Engineering (jee.ro). 13.,* pp. 301-307, Januar 2014.

[36] M. V. Iordache und P. J. Antsaklis, „SPNBOX - A Toolbox for the Supervisory Control of Petri Nets," 14 Mai 2013. [Online]. Available: http://mviordache.name/abs/spnbox/. [Zugriff am 12 Mai 2018].

[37] Q. Shafiee, T. Dragicevic, J. C. V. Quintero und J. M. Guerrero, „Hierarchical Control for Multiple DC-Microgrids Clusters," *Proceedings of the 11th International Multi-Conference on Systems, Signals and Devices, SSD 2014,* 2014.

[38] A. Brocco, „Fully distributed power routing for an ad hoc nanogrid," in *2013 IEEE International Workshop on Inteligent Energy Systems (IWIES)*, Wien, Österreich, 2013.

A. Anhang

Der Anhang dieser Masterarbeit befindet sich auf der beiliegenden CD, die beim Erst- und Zweitprüfer eingesehen werden kann.

A.1 DESTool Projekte

Im Ordner „\DESTool_Projekte“ sind die entwickelten Modelldateien im Dateiformat „.pro“ hinterlegt, welche mit der Entwicklungsumgebung DESTool geöffnet werden können.

Es ist jeweils eine Datei für den

- monolithischen Steuerungsentwurf: „\Modell_nanogrid_monolithisch.pro“ und
- lokal-modularen Steuerungsentwurf: „\Modell_nanogrid_lokal-modular.pro“

abgelegt.

Die geschlossenen Steuerkreise in beiden Dateien sind bereits alle mit dem Attribut *Plant* versehen. Damit können direkt im Animationsreiter die nicht steuerbaren Ereignisse zur Verifikation der durchgeführten Simulationsschritte angesteuert werden

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

_____ _____

Ort, Datum Unterschrift