Variantenmanagement in der Modellbildung und Simulation unter Verwendung des SES/MB Frameworks

Dissertation

zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von

Artur Schmidt, geboren in Pawlowka

Rostock, im Oktober 2018

https://doi.org/10.18453/rosdok_id00002492

Eingereicht am: 22. Oktober 2018

Verteidigt am: 11. Juni 2019

Gutachter: Prof. Dr.-Ing. Torsten Jeinsch
Institut für Automatisierungstechnik, Universität Rostock

Prof. Dr. techn. Felix Breitenecker
Institut für Analysis und Scientific Computing, TU Wien

Prof. Dr.-Ing. Thorsten Pawletta
FG Computational Engineering & Automation, Hochschule Wismar

Kurzfassung

Eine wesentliche Grundlage des modernen Entwicklungs- und Planungsprozesses in den Ingenieurwissenschaften bildet die Methode der Modellbildung und Simulation (M&S). Die Modellbildung beschäftigt sich mit der Abstraktion wesentlicher Aspekte technischer Systeme und deren Umsetzung in einem Modell. Mit der Simulation werden die erstellten Modelle auf einem Computer ausgeführt, mit dem Ziel Erkenntnisse über das Systemverhalten zu gewinnen. Die einmalige Ausführung eines Modells wird als Simulationslauf bezeichnet. Ein einfaches Simulationsexperiment umfasst eine Menge von Simulationsläufen mit variierten Einflussgrößen. Komplexe simulationsbasierte Experimente sind durch die Integration der Simulation mit anderen numerischen Verfahren, wie zum Beispiel der Optimierung, dem Screening oder der Sensitivitätsanalyse, gekennzeichnet. Bei hoch komplexen Simulationsexperimenten ist neben dem Parameterraum auch die Modellstruktur Gegenstand der Untersuchung.

Die klassische Vorgehensweise der M&S stößt im Ingenieurbereich zunehmend an ihre Grenzen. Ein Grund sind wachsende Ansprüche von Kunden nach individuellen Produkten. Dadurch steigen die Variantenvielfalt und oft die Komplexität technischer Systeme und damit auch der daraus resultierenden Modelle sowie Untersuchungsziele. Das heißt, die Spezifikation und Ausführung von Simulationsexperimenten umfasst nicht nur eine Menge von Modellvarianten mit zulässigen Parametrierungen, sondern auch Kombinationen numerischer Verfahren und deren Konfiguration. Weiterhin ergibt sich die Folge zu untersuchender Varianten oder anzuwendender numerischer Methoden oft erst im Verlauf eines Experiments aus zuvor ermittelten Untersuchungsergebnissen.

Die vorliegende Arbeit leistet demgemäß einen Beitrag zur Entwicklung von allgemeinen Methoden zum Variantenmanagement in der M&S bis auf die Ebene der Simulationsexperimente und zur automatisierten Ausführung von Simulationsexperimenten. Die Lösungsvorschläge werden aufbauend auf dem ASIM-Vorgehensmodell, der modular-hierarchischen Modellbildung und dem System Entity Structure/Model Base (SES/MB) Framework entwickelt. Das Variantenmanagement basiert auf verschiedenen Phasen, der *Variantenanalyse*, *-formalisierung*, *-implementierung* und *-generierung*. Weiterhin wird eine modular-hierarchische Strukturierung für Simulationsexperimente entwickelt. Durch eine Erweiterung des SES/MB Frameworks können nicht nur Modellvarianten, sondern vollständige Experimentvarianten spezifiziert, implementiert und generiert werden. Weiterhin wird ein Konzeptrahmen zum automatisierten Experimentieren basierend auf dem erweiterten SES/MB Framework erarbeitet. Abschließend werden die entwickelten Methoden an einem Anwendungsbeispiel aus dem Bereich der Produktion und Logistik präsentiert.

Abstract

Modeling and Simulation (M&S) forms the basis of the modern development and planning process in engineering. Modeling deals with the abstraction of essential aspects of technical systems and their implementation in a model. During simulation the implemented model is executed aiming to gain knowledge about the system behavior. A single model execution is called a simulation run. A simple simulation experiment comprises many simulation runs with varied input variables. Complex simulation-based experiments are characterized by the integration of simulation with numerical methods such as optimization, screening or sensitivity analysis. Highly complex simulation experiments investigate the different model structures in addition to the parameter space.

The classical M&S approach is reaching its limits in engineering. One reason is the growing customers demand for customized products. This increases the variety and often the complexity of technical systems as well as of the resulting models and investigation objectives. This means that the specification and execution of simulation experiments includes not only a set of model variants with their parameter settings, but also combinations with numerical methods and their configuration. Furthermore, the sequence of variants to be investigated or numerical methods to be applied often arise from previously determined experiment results.

This thesis contributes to the development of general methods for variant management in M&S up to the simulation experiments' level and their automated execution. The proposed solutions are developed based on the ASIM procedure model, the modular-hierarchical modeling approach and the System Entity Structure/Model Base (SES/MB) framework. The variant management comprises the phases *variant analysis*, *variant formalization*, *variant implementation* and *variant generation*. Furthermore, a modular-hierarchical structuring for simulation experiments is presented. By extending the SES/MB framework, not only model variants but complete experiment variants can be specified, implemented and generated. Moreover, a framework for automated experimentation based on the extended SES/MB framework is proposed. Finally, the methods are demonstrated using an application example from the field of production and logistics.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ALD	Außenlängsdrehen
ARS	Außenrundschleifen
ASIM	Arbeitsgemeinschaft Simulation
AVES	Air VEhicle Simulator
AXi	i-tes Axiom
CNC	Computerized Numerical Control
DEVS	Discrete EVent System Specification
DOE	Design of Experiment
DSL	Domain Specific Language
EF	Experimental Frame
EM	Experimentmethode
FIFO	First In First Out
FM	Featuremodell
FOD	Feature Oriented Development
FOMDD	Feature Oriented Model Driven Development
G	Generator
GA	Genetischer Algorithmus
GM	Grinding Machine
GUISE	GUIding Simulation Experiments
HE	Hierarchieebene
HF	Hardening Furnace
IH	Induktionshärten
LIFO	Last In First Out
LP	Logischer Prozess
M&S	Modellbildung und Simulation
MATLAB	MATrix LABoratory
MB	Model Base
MDD	Model Driven Development

MUS	Model Under Study
OG	Obergrenze
P&L	Produktion und Logistik
PDEVS	Parallel Discrete EVent System Specification
PES	Pruned Entity Structure
RBD	Random Balance Design
SA	Sensitivitätsanalyse
SC	Screening
SES	System Entity Structure
SESVars	SES Variablen
SimM	Simulationsmodell
SLB	Sobol
SM	Simulationsmethode
SPO	Simulationsbasierte Optimierung
SSPO	Simulationsbasierte Struktur und Parameter Optimierung
T	Senke
TF	Tempering Furnace
TM	Turning Machine
UG	Untergrenze
V&V	Verifikation und Validierung
VDI	Verein Deutscher Ingenieure
VH	Vakuumhärten
VP	Variationspunkt
VV&T	Verifikation, Validierung und Testing

1 Einleitung

Die Methode der *Modellbildung und Simulation* (M&S) findet heute in nahezu jedem Wissenschaftsbereich ihre Anwendung. Speziell in den Ingenieurwissenschaften ist die M&S zu einem wesentlichen Bestandteil bei Entwicklungs- und Planungsprozessen geworden. Die Grundidee der M&S basiert darauf, ein Systemverhalten anhand eines Modells zu prognostizieren. Zunächst werden mittels der *Modellbildung* die wesentlichen Aspekte eines Systems in einem Modell abgebildet. Das Modell ist eine abstrakte und damit vereinfachte Repräsentation des Systems. Zur Steigerung der Übersicht kommen in den Ingenieurwissenschaften meist modular-hierarchische Modelle zum Einsatz. Modular-hierarchische Modelle bestehen aus Modulen, Komponenten oder Teilmodellen, die sowohl horizontal als auch hierarchisch miteinander verknüpft sind. Die Prognose des Systemverhaltens erfolgt im nächsten Schritt mit der *Simulation*, worunter die Abarbeitung oder Ausführung des Modells unter Verwendung eines Simulators verstanden wird. Die einmalige Ausführung eines Modells durch einen Simulator nennt man Simulationslauf und stellt die Grundlage von Simulationsexperimenten dar. Unter einem Simulationsexperiment wird, wie im Allgemeinen in der Wissenschaft, eine methodisch angelegte Untersuchung verstanden, bei welcher Einflussgrößen verändert werden. Die Variation der Einflussgrößen kann manuell durch den Nutzer oder durch einen Algorithmus, also durch eine Softwarekomponente, erfolgen. Typische Simulationsexperimente sind: simulationsbasierte Parameterstudien, die simulationsbasierte Optimierung, simulationsbasiertes Screening oder die simulationsbasierte Sensitivitätsanalyse. Das Attribut simulationsbasiert kennzeichnet in diesem Kontext ein Experiment mit integriertem Simulator. Die Interpretation gewonnener Ergebnisse ermöglicht je nach Experiment und Zielstellung unterschiedliche Rückschlüsse zum Systemverhalten.

Die klassische Vorgehensweise der M&S stößt im Ingenieurbereich bei immer mehr Anwendungen an ihre Grenzen. Ein Grund sind wachsende Kundenansprüche nach individuellen Produkten. Klassisches Beispiel hierfür ist das Automobil. Um den Ansprüchen der Kunden gerecht zu werden, steigt die Fahrzeug- und Funktionsvielfalt sukzessive an und führt somit zu vielfältigen Fahrzeugtypen und -ausführungen. Demgemäß beschränkt sich der Entwicklungs- und Produktionsprozess nicht mehr nur auf *ein oder wenige* Systeme. Weiterhin kommen immer neue Anforderungen hinzu, die die Systemvariabilität und Komplexität erhöhen. Während beim Produktionsprozess in früheren Zeiten insbesondere die Auslastung von Anlagen sowie Termineinhaltungen im Vordergrund standen, kommt heute zum Beispiel oft die Berücksichtigung energetischer Aspekte hinzu. Aus der Varianten- und Produktionsvielfalt folgt eine Vielzahl unterschiedlicher Systeme. Die Beherrschung dieser Vielfalt und der wachsenden Komplexität erfordert umso mehr den Einsatz von M&S Methoden sowie deren methodische Weiterentwicklung.

Mit der Variantenvielfalt sowie der Komplexität der Modelle und Untersuchungsziele steigt gleichermaßen die Komplexität simulationsbasierter Experimente enorm an. Oft ist es schwierig für simulationsbasierte Experimente passende numerische Verfahren sowie deren Parametrierung auszuwählen. Ein typisches Beispiel ist die simulationsbasierte Optimierung. Es existieren unterschiedliche Optimierungsmethoden, zumeist mit einer immensen Menge an Konfigurationsmöglichkeiten. Der Erfolg einer simulationsbasierten

Optimierung hängt oft stark von der ausgewählten Methode und deren Konfiguration ab. Gleiches gilt auch für das simulationsbasierte Screening und Sensitivitätsanalysen. Die Variantenvielfalt potenziert sich durch den kombinierten Einsatz von numerischen Verfahren im Rahmen eines simulationsbasierten Experiments.

Ein Anwender muss bei der Planung komplexer Simulationsstudien nicht nur eine Menge unterschiedlicher Modellvarianten und deren zulässige Parametrierungen spezifizieren, sondern auch die dazu notwendigen Kombinationen numerischer Methoden und deren Konfigurationsparameter. Erschwerend kommt hinzu, dass die Folge zu untersuchender Modellvarianten sowie die Auswahl und Parametrierung der numerischen Methoden sich oft erst im Verlaufe eines Experiments auf Basis von Untersuchungsergebnissen ergeben. Das bisher vorherrschende manuelle Experimentieren auf dieser Komplexitätsebene bedeutet, hochqualifiziertes Personal mit einer monotonen, ermüdenden sowie zeitaufwendigen Tätigkeit zu binden. Demgemäß sollten Ansätze zur Automatisierung des simulationsbasierten Experimentierens gesucht werden. In der M&S Theorie existieren erste Ansätze zur besseren Beherrschung der Variantenvielfalt. Jedoch wird die Variantenvielfalt bisher nahezu ausschließlich auf der Seite der Modellbildung betrachtet. Eine umfassende Betrachtung bis auf die Ebene des simulationsbasierten Experimentierens erfolgte bisher kaum. Es fehlen Methoden zur ganzheitlichen Spezifikation der Variantenvielfalt und der automatisierten Ausführung von komplexen Simulationsstudien. In Anlehnung an die Begriffsbildung im Software Engineering wird im Weiteren die Beherrschung von Variantenvielfalt mit dem Begriff *Variantenmanagement* gleichgesetzt.

Die allgemeine Zielstellung dieser Arbeit ist das Variantenmanagement im Bereich der M&S mit einem erweiterten Blick auf durchzuführende simulationsbasierte Experimente. Dabei wird das Variantenmanagement hinsichtlich der Modelle nur untergeordnet behandelt und maßgeblich auf bereits existierende Methoden aufgebaut. Die Grundlagen der zu entwickelnden Methodik bilden das im deutschen Sprachraum verbreitete *ASIM-Vorgehensmodell* der M&S und das aus der Systemtheorie abgeleitete *System Entity Structure/Model Base Framework* (SES/MB). Ersteres fungiert als allgemeiner Vorgehensplan bezüglich der einzelnen Phasen und der zugehörigen Phasenergebnisse. Das SES/MB Framework ermöglicht eine abstrakte Spezifikation einer Menge modular-hierarchischer Modelle und definiert abstrakte Methoden zur Generierung ausführbarer Modelle. Die MB entspricht einer Modellbibliothek, welche eine Menge konfigurierbarer Modellkomponenten mit eindeutigen Ein- und Ausgangsschnittstellen organisiert. Die SES ist ein gerichteter azyklischer Graph und spezifiziert eine Menge modular-hierarchischer Modellstrukturen, Modellparametrierungen sowie Links zu den Modellkomponenten in der MB. Für die Spezifikation und Generierung ausführbarer Simulationsexperimente mit dem SES/MB Framework muss analog zur Modellebene eine modular-hierarchische Strukturierung für Simulationsexperimente erarbeitet werden. Dazu soll auf das Konzept des *Experimental Frame* aufgebaut werden. Im nächsten Schritt soll ein *Konzeptrahmen für ein automatisiertes Experimentieren* mit unterschiedlichen Varianten simulationsbasierter Experimente erarbeitet werden. Der Konzeptrahmen soll den Anwender bei der zeitaufwändigen Routinearbeit zur Durchführung einer Vielzahl unterschiedlicher Simulationsexperimente entlasten. Nachfolgend wird die Struktur der Arbeit kurz beschrieben.

Das zweite Kapitel beschäftigt sich, ausgehend vom ASIM-Vorgehensmodell, mit den Grundlagen der klassischen M&S. Dabei werden die einzelnen Phasen analysiert und bereits ausgewählte Ansätze zum Variantenmanagement für die Modellebene betrachtet.

Im dritten Kapitel wird das SES/MB Framework eingeführt. Es werden die Grundlagen, offene Probleme und notwendige Lösungsvorschläge für den Einsatz des SES/MB Framework als Basis für ein Variantenmanagement in der M&S diskutiert. Anschließend wird aufbauend auf den Ergebnissen des zweiten Kapitels ein erweitertes Vorgehensmodell mit Berücksichtigung eines Variantenmanagements für die Modellebene vorgestellt.

Ausgehend vom SES/MB Framework, als Basis für ein Variantenmanagement simulationsbasierter Experimente, wird im Kapitel vier eine modular-hierarchische Strukturierung für simulationsbasierte Experimente entwickelt. Dies erfolgt aufbauend auf dem Konzept des *Experimental Frame*.

Kapitel fünf beschäftigt sich mit der Entwicklung eines Konzeptrahmens für das automatisierte Experimentieren. Dieser baut auf dem entwickelten Variantenmanagement mit dem SES/MB Framework auf. Der Konzeptrahmen ermöglicht es, eine Vielzahl unterschiedlicher Simulationsexperimente auf Basis einer Anwenderspezifikation automatisiert zu generieren und auszuführen. Zur Reduktion des Laufzeitaufwands bei rechenzeitintensiven Experimenten wird diskutiert, wie auf Methoden aus dem Bereich des *Parallen und Verteilten Rechnens* zurückgegriffen werden kann.

Eine detaillierte Demonstration der entwickelten Methodik erfolgt im sechsten Kapitel an einem Anwendungsbeispiel aus dem Bereich der Produktion und Logistik. Als Simulationsumgebung wird MATLAB/SimEvents verwendet.

Abschließend werden die erzielten Ergebnisse zusammengefasst und ein Ausblick auf mögliche weiterführende Arbeiten gegeben.

2 Grundlagen und Variantenvielfalt der Modellbildung und Simulation

In diesem Kapitel werden zunächst wichtige Begriffe der klassischen Modellbildung und Simulation eingeführt und erläutert. Danach wird der Fokus auf simulationsbasierte Experimente gelegt. Es wird eine allgemeine Klassifikation eingeführt und der Aufbau und Ablauf von Experimenten diskutiert. Anschließend folgt eine Einführung in das Gebiet des Variantenmanagements im Bereich der Modellbildung und Simulation (M&S). Das Kapitel schließt mit einer Zusammenfassung und Diskussion der noch offenen Probleme in der M&S.

2.1 Grundlagen der klassischen Modellbildung und Simulation

Die Methode der *Modellbildung und Simulation* (M&S) findet fast in jedem Wissenschaftsbereich ihre Anwendung. Nach Glöckler [50] bildet die M&S heutzutage speziell in den Ingenieurwissenschaften einen wichtigen Teil des modernen Entwicklungsprozesses. Die allgemeine Grundidee basiert darauf, ein Systemverhalten ohne empirische Untersuchung des Systems, sondern anhand eines zugehörigen Modells zu prognostizieren [135]. Im ersten Schritt werden unter Verwendung der *Modellbildung* nur die wesentlichen Aspekte eines Systems, die zur Lösung einer bestimmten Problemstellung notwendig sind, in einem Modell nachgebildet. Ein Modell kann somit gemäß Mittal und Matin [102] als eine abstrakte und damit vereinfachte Repräsentation eines Systems definiert werden. Für die Prognose des Systemverhaltens erfolgt im nächsten Schritt die *Simulation.* Diese wird in Anlehnung an Cellier und Kofman [18] als das Ausführen von Experimenten am Modell definiert. Die Analyse der gewonnenen Simulationsergebnisse ermöglicht den Rückschluss auf das Systemverhalten.

Die Methode M&S verdrängt in der heutigen Welt sukzessiv das Experimentieren am System. Zur Begründung dieser These führt Sauerbier [135] mehrere Punkte als Vorteile für die Simulationsexperimente auf:

- Das System ist nicht beziehungsweise noch nicht vorhanden.
- Das System ist nicht direkt beeinflussbar.
- Das Experiment ist zu gefährlich oder bedenklich.
- Das Experiment ist zu teuer.
- Das Experiment führt zur Zerstörung des realen Systems.
- Die zukünftige oder vergangene Entwicklung des Systems soll analysiert werden.
- Zu untersuchende Zusammenhänge sind am System nicht oder begrenzt beobachtbar.

Es ist jedoch auch klar, dass neben den vielen Vorteilen auch Nachteile für den Einsatz der Simulation existieren. Ein wesentlicher Nachteil der Simulation ist, dass die erzielten Simulationsergebnisse nie exakt der Realität entsprechen und diese auch nie als solche behandelt werden dürfen. Denn wie bereits einleitend erwähnt, stellt ein Simulationsmodell ausschließlich eine abstrakte Darstellung des Systems dar. Somit stellt der Vorgang der Interpretation der Simulationsergebnisse einen essentiellen Aspekt der M&S dar.

Nach der Aussage von Sauerbier [135] sind, bei der Durchführung von M&S-Projekten, die einzelnen Tätigkeiten nach einem vorgegebenen Ablauf aneinandergereiht. Solche Abläufe werden Vorgehens- oder Lebenszyklusmodelle genannt und fungieren allgemein als Orientierungsplan und visualisieren die einzelnen durchzuführenden Phasen und die zugehörigen Phasenergebnisse eines M&S-Projektes. In der Literatur existieren unterschiedliche Vorgehens- und Lebenszyklusmodelle wie zum Beispiel in [5, 134, 124, 8]. Eine Übersicht und nähere Betrachtung der jeweiligen Vorgehens- und Lebenszyklusmodelle erfolgt beispielsweise durch Rybaki [129] sowie Rabe et al. [124]. Das im deutschen Sprachraum anerkannte Vorgehensmodell der Arbeitsgemeinschaft Simulation (ASIM) nach Rabe et al. [124] ist Bestandteil der VDI-Richtlinie 3633 [171] und in Abbildung 1 zu sehen.

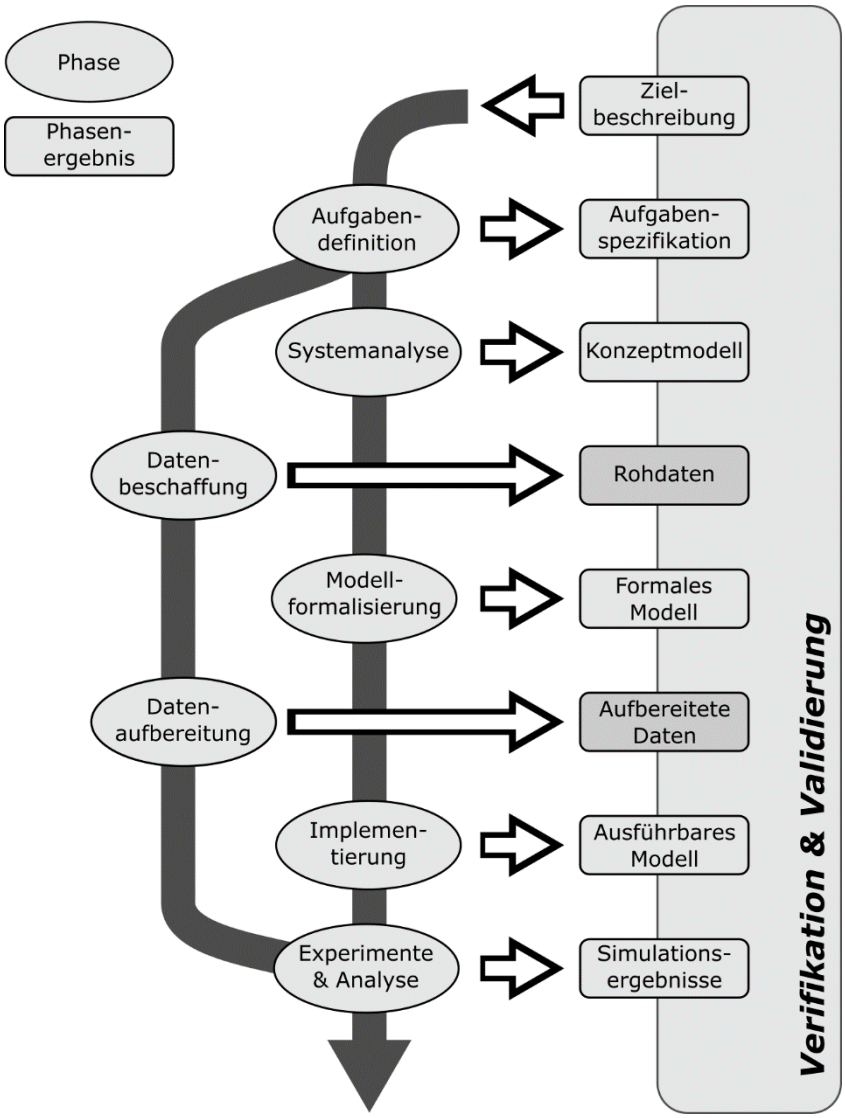


Abbildung 1: ASIM-Vorgehensmodell zur Durchführung von M&S-Projekten nach [124, 171].

Die durchzuführenden Phasen sind mit Ellipsen und die Phasenergebnisse als Vierecke markiert. Jedoch entspricht die *Zielbeschreibung* nach Rabe et al. keinem Ergebnis, sondern bildet die Ausgangsbasis eines M&S-Projektes. Gemäß Wenzel et al. [178] umfasst die *Modellbildung* die drei Phasen (i) *Systemanalyse*, (ii) *Modellformalisierung* sowie (iii) *Implementierung* und beinhaltet die phasenbezogenen Ergebnisse (i) *Konzeptmodell*, (ii)

Formales Modell und (iii) *Ausführbares Modell*. Die *Simulation* entspricht der Phase *Experimente & Analyse* mit dem Phasenergebnis *Simulationsergebnisse*. Nach den Angaben von Rabe et al. [124] sind die Phasen *Datenbeschaffung* und *Datenaufbereitung* mit den zugehörigen Phasenergebnissen *Rohdaten* und *Aufbereitete Daten* bewusst aus dem Ablauf der Modellbildung herausgenommen, da ihre Durchführung sowohl inhaltlich als auch zeitlich unabhängig von der Modellbildung erfolgen kann. Zur Steigerung der Qualität eines M&S-Projektes ist, wie in Abbildung 1 gezeigt, eine projektbegleitende sowie durchgängige *Verifikation* und *Validierung* (V&V) der Phasenergebnisse notwendig [178]. Balci [5, 6] spricht in diesem Zusammenhang auch von *Validierung*, *Verifikation* und *Testing* (VV&T) und weist darauf hin, dass diese Begriffe nicht unabhängig voneinander betrachtet werden können. Demnach bildet das Testing die Grundlage für die Verifikation und Validierung. In dieser Arbeit wird der Logik von Balci gefolgt und im weiteren Verlauf der Begriff VV&T verwendet. Zur Definition der drei Begriffe lehnt sich die vorliegende Arbeit an die von Balci [5] eingeführten Definitionen an. Die Validierung geht der Fragestellung nach, ob das richtige Modell erstellt wurde, und die Verifikation, ob das Modell richtig erstellt wurde. Testing hingegen definiert Balci als einen Vorgang zur systematischen Ermittlung von Ungenauigkeiten oder Fehlern eines Modells. Somit muss zunächst das aktuelle Phasenergebnis *positiv* abgeschlossen sein, bevor die nächste Phase abgearbeitet werden kann.

Anders als in Abbildung 1 dargestellt, argumentiert Liebl [93], dass die einzelnen Phasen des Vorgehensmodells weder einmalig durchlaufen, noch in jedem Fall einem linearen Schema folgen. Vielmehr wird ein M&S-Projekt als ein iterativer Vorgang angesehen, bei dem einzelne Phasen mehrmals durchlaufen werden können.

Zur besseren Abgrenzung zu anderen Arbeiten soll an dieser Stelle eine Einordnung der vorliegenden Arbeit nach dem Vorgehensmodell in Abbildung 1 erfolgen. Die vorliegende Arbeit behandelt ausschließlich die Phasen der Modellbildung und Simulation. Bei der Modellbildung erfolgt eine Fokussierung auf Aspekte der modular-hierarchischen Modellbildung, auch kompositionale Modellbildung [45] oder signalflussorientierte Methode [189] genannt. Weiterhin werden im Rahmen dieser Arbeit die Phasen *Datenerfassung*, *Datenaufbereitung* und *Modellformalisierung* sowie die VV&T Thematik explizit ausgeklammert, da ihre Mitbetrachtung den Rahmen der Arbeit sprengen würde. Prinzipiell sollte aber die entwickelte Methodik des Variantenmanagements auch eine Integration dieser Schwerpunkte unterstützen.

Im weiteren Verlauf werden die Phasen der Modellbildung und Simulation näher beleuchtet.

2.1.1 Systemanalyse als Ausgangspunkt der Modellbildung

Der Begriff *System* ist in der Literatur weit gefasst und unterschiedlich definiert. Im Rahmen der Ingenieurwissenschaften definiert Glöckler [50] ein System als eine Gesamtheit von Elementen, die zusammengenommen eine zweckgebundene Einheit ergeben und sich durch eine Systemgrenze von der Umgebung abgrenzen lassen. Die Umgebung entspricht einer Menge von Elementen, die nicht Teil des Systems sind, jedoch das Verhalten des Systems beeinflussen können. Einzelne Systemelemente interagieren über Schnittstellen miteinander

und mit deren Umgebung. Weiterhin besitzt jedes System eine Menge von Einflussgrößen (X), Ausgabegrößen (Y) und Systemzuständen (S) [97]. Zeigler et al. [187] fordern eine strikte Trennung zwischen der Systemdynamik und der Systemstruktur. Dabei steht die Systemdynamik in einem direkten Zusammenhang mit dem Systemverhalten, also mit der äußeren Manifestation eines Systems. Die Systemstruktur beschreibt die innere Zusammensetzung eines Systems. Im Kontext dieser Arbeit wird der Begriff System auf *dynamische Systeme* angewendet. Ein wesentliches Charakteristikum dynamischer Systeme ist die explizite Berücksichtigung der Zeit [29]. Das heißt, die Einflussgrößen, Ausgabegrößen und die Systemzustände unterliegen zeitlichen Änderungen. Ein weiteres Merkmal ist die Rückwirkungsfreiheit dynamischer Systeme [97]. Zur Minimierung des Schreibaufwandes werden die Begriffe System und dynamisches System im Weiteren synonym verwendet. Abbildung 2a zeigt die wesentlichen Elemente eines dynamischen Systems. Für ein besseres Verständnis und Erfassung von Wirkzusammenhängen von Systemen hat sich die Darstellung mittels Blockschaltbildern durchgesetzt. Blockschaltbilder zählen zu den graphischen Modellierungssprachen. Zander [183] argumentiert mit Bezug auf [65, 73], dass die Verwendung graphischer Modellierungssprachen zu einer vier- bis zehfachen Produktivitätssteigerung führen kann. Den Grund hierfür liefern Roßner et al. [127] mit der Aussage, dass das menschliche Gehirn visuelle Informationen effizienter aufnimmt als textuelle. Abbildung 2b zeigt ein Blockschaltbild einer Systemstruktur mit unterschiedlichen Hierarchieebenen.

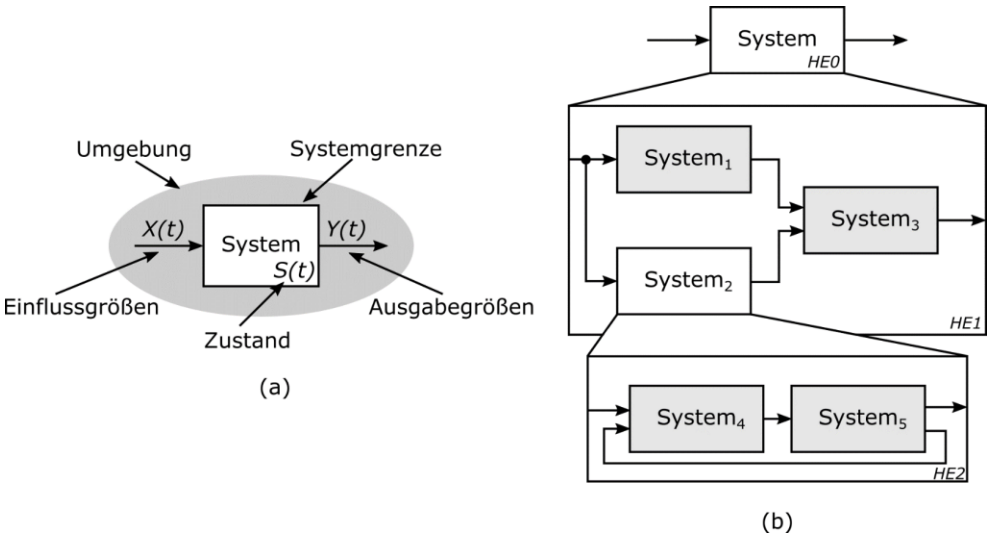


Abbildung 2: (a) Elemente eines Systems mit Umgebung. (b) Blockschaltbild eines Systems mit unterschiedlichen Hierarchieebenen (HE).

Blöcke kennzeichnen Systeme und stellen gleichzeitig deren Systemgrenze dar. Pfeile kennzeichnen die Relationen zwischen den einzelnen Systemen. Der Richtungssinn gibt Aufschluss über die Wirkzusammenhänge. Ist der Wirkzusammenhang eines Blockschaltbildes eindeutig erkennbar, spricht man von einem kausalen Blockschaltbild, anderenfalls von einem akausalen Blockschaltbild. In dieser Arbeit werden ausschließlich kausale und rückwirkungsfreie Systeme betrachtet.

Bei der Systemanalyse müssen, in Anlehnung an Wenzel et al. [178], folgende Informationen ermittelt werden:

- Bestimmung der Systemgrenze sowie Möglichkeiten der Zerlegung des Systems und Ableitung daraus folgender Systemstrukturen,
- Bestimmung relevanter Systemelemente mit zugehörigen Systemgrenzen,
- Bestimmung der Einflussgrößen, Ausgabegrößen sowie Systemzustände,
- Bestimmung der Relationen zwischen den Systemelementen sowie
- Bestimmung der zu erfassenden Informationen und Daten.

Balzert [7] beschreibt acht Prinzipien als wesentliche Basistechniken der Systemanalyse in der Softwaretechnik: (i) *Prinzip der Abstraktion*, (ii) *Prinzip der Strukturierung*, (iii) *Prinzip der Bindung und Kopplung*, (iv) *Prinzip der Hierarchisierung*, (v) *Prinzip der Modularisierung*, (vi) *Geheimnisprinzip*, (vii) *Prinzip der Lokalität* sowie das (viii) *Prinzip der Verbalisierung*. Die Ergebnisse der Systemanalyse werden im Konzeptmodell zusammengefast. Zur formalen Erstellung des Konzeptmodells können beispielsweise die *Unified Modeling Language* [7] oder die *System Modeling Language* [45] herangezogen werden.

2.1.2 Formalisierung und Implementierung des Simulationsmodells

Ziel der Systemanalyse ist nach Abbildung 1 die Schaffung eines Konzeptmodells als notwendige Grundlage für die anschließenden Phasen der Formalisierung und Implementierung. Das Konzeptmodell ist ein rein deskriptives, abstraktes und informelles Modell des zu untersuchenden Systems. Durch den Arbeitsschritt der *Modellformalisierung* werden die im Konzeptmodell beschriebenen Sachverhalte konkretisiert und formalisiert.

Der zur Formalisierung genutzte Formalismus hängt vom abzubildenden Modellverhalten ab. Grundlegend können Modelle gemäß ihrem Verhalten in kontinuierlich, zeitdiskret, ereignisdiskret und hybrid eingeteilt werden. Mögliche Formalismen sind zum Beispiel gewöhnliche Differentialgleichungen, Bondgraphen [14], Petrinetze [125], der Harel-Automat [63], die Discrete Event System Specification (DEVS) [187] und so weiter. Die später im Anwendungsteil betrachteten Beispielprobleme aus dem Themengebiet der Produktion und Logistik werden nach Eley [37] der ereignisdiskreten M&S zugeordnet. Tabelle 1 zeigt stellvertretend den auf DEVS basierenden PDEVS-Formalismus zur formalen Spezifikation ereignisdiskreter Modelle, der strikt zwischen Modelldynamik (atomares Modell) und Modellstruktur (gekoppeltes Modell) unterscheidet. Gemäß der klassischen M&S Theorie, wie zum Beispiel in Zeigler et al. [187], sollte das formale Modell keine softwarespezifischen Aspekte beinhalten, sondern nur die zu modellierende Problemstellung abstrakt und formal abbilden. Wenzel et al. [178] stellen diesen strikten Ansatz der Formalisierung in Frage, da mit der Formalisierung ein erheblicher Zeitaufwand einhergeht. Erfolgen keine formalen Modellanalysen, wie beispielsweise Erreichbarkeits- oder Deadlock-Analysen, ist der Vorteil der strikten Formalisierung bei Verwendung moderner Modellierungs-/Simulationsumgebungen meist nicht mehr begründbar. Die Softwaresysteme bieten in der Regel zwar nur softwareabhängige Formalisierungskonzepte, unterstützen dafür aber einen durchgängigen Übergang zur Implementierung, wodurch sich erhebliche Zeiteinsparungen ergeben. Diesen Argumenten folgend, wird in dieser Arbeit die Phase der Formalisierung nicht explizit betrachtet.

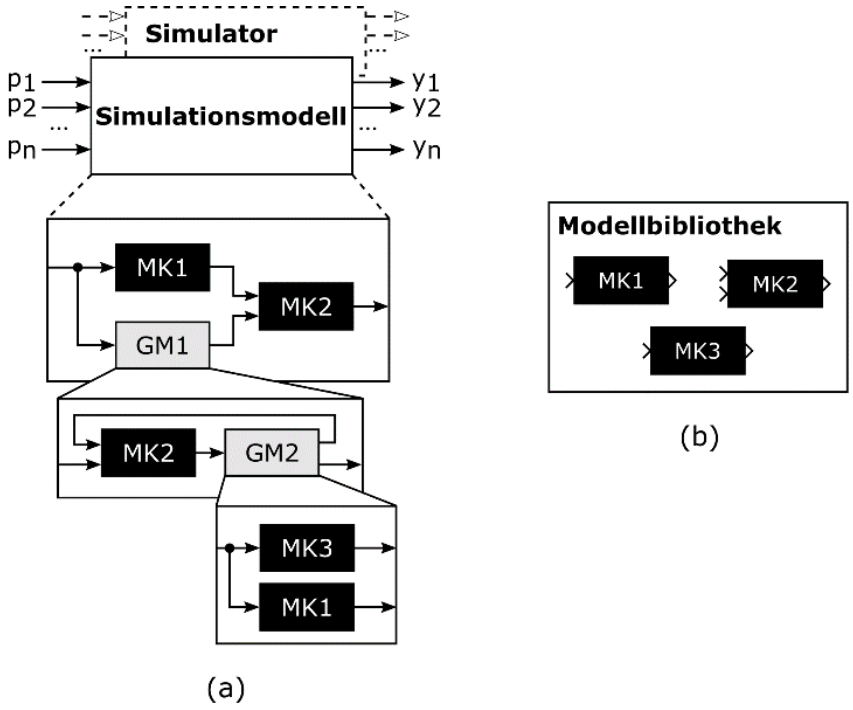
Tabelle 1: PDEVS Spezifikation atomarer und gekoppelter Modelle nach [187].

Atomares Modell	**Gekoppeltes Modell**
$PDEVS_{atomic} = (X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda)$	$PDEVS_{coupled} = (X, Y, D, \{M_d\}, IC, EIC, EOC)$
X Menge der Eingangsgrößen	X Menge der Eingangsgrößen
Y Menge der Ausgabegrößen	Y Menge der Ausgabegrößen
S Menge der Zustände	D Menge der Komponentennamen
δ_{int}: S→S interne Zustandsüberführungsfunktion	M_d Menge der PDEVS Komponenten, d ϵ D
δ_{con}: S×X →S konfluente Zustandsüberführungsfunktion	IC Menge der internen Kopplungen
δ_{ext}: Q×X→S externe Zustandsüberführungsfunktion, mit $Q=\{(s,e)\|s \epsilon S, 0 \leq e \leq ta(s)\}$ totale Zustandsmenge; e entspricht der verstrichenen Zeit seit dem letzten Ereignis	
λ: S→Y Ausgabefunktion	EIC Menge der externen Eingangskopplungen
ta: S→$R^{+}_{0,\infty}$ Zeitfortschrittsfunktion	EOC Menge der externen Ausgangskopplungen

Das strikte formale Modell ist computerunabhängig und somit auch nicht ablauffähig. In der Phase *Implementierung* wird das formale Modell unter Verwendung einer allgemeinen Programmiersprache oder einer Domain Specific Language (DSL), einer *Mini-Sprache* für einen spezifischen Anwendungsbereich [27, 24], softwaretechnisch in einer Simulationsumgebung implementiert. Das Ergebnis der Implementierung entspricht nach Abbildung 1 einem *Ausführbaren Modell*, welches in dieser Arbeit, gemäß Uhrmacher und Weyns [167], als Simulationsmodell bezeichnet wird.

Zur Beherrschung der Komplexität und der Vermeidung monolithischer Simulationsmodelle erfolgt, beginnend mit der Systemanalyse bis zur Implementierung des Simulationsmodells, eine modular-hierarchische Herangehensweise gemäß dem Prinzip der Modularisierung und Hierarchisierung nach Balzert [7]. Demnach besteht ein Simulationsmodell aus einer Menge hierarchisch angeordneter, miteinander gekoppelter, in sich geschlossener und austauschbarer Module. Durch die Hierarchisierung erfolgt eine Beschränkung auf definierte, gerichtete Strukturen, um so chaotische Strukturen zu vermeiden. Zupancic [190] definiert den Begriff Modul als eine wiederverwendbare unabhängig agierende funktionale Komponente. Demnach kann ein Modul mit einer (Modell-)Komponente gleichgesetzt werden. Weiterhin wird eine Komponente als eine wiederverwendbare Black-Box mit wohldefinierten Ein-/Ausgabeschnittstellen aufgefasst werden, welche in einer Modellbibliothek organisierbar sind. Der Begriff Black-Box soll verdeutlichen, dass weder die interne Dynamik noch die strukturelle Zusammensetzung eine übergeordnete Rolle einnehmen. Ein weiterer wesentlicher Aspekt bei der Implementierung ist die konsequente Trennung zwischen Daten und Simulationsmodell [178]. Eine solche Trennung trägt zur Transparenz des Simulationsmodells bei und ermöglich eine bessere Variation und Wartung von Simulationsmodellen.

Zur Abarbeitung eines Simulationsmodells wird ein Simulator benötigt. Ein Simulator ist allgemein gesehen ein Computeralgorithmus für die Abarbeitung eines Simulationsmodells und bildet einen essentiellen Bestandteil jeder Simulationsumgebung. Abbildung 3 zeigt schematisch eine Modellbibliothek und ein darauf basierendes modular-hierarchisches Simulationsmodell mit zugeordnetem Simulator.



MK...Modellkomponente; GM...Gekoppeltes Modell; p...Einflussgröße; y...Ausgabegröße

Abbildung 3: (a) Modular-hierarchisches Simulationsmodell mit zugehörigem Simulator und (b) zugehöriger Modellbibliothek.

Nachfolgend werden die Ein- und Ausgänge eines Simulationsmodells, analog zu Systemen, Einflussgrößen und Ausgabegrößen genannt. Oft werden die Einflussgrößen eines Simulationsmodells auch als Parameter und die Ausgabegrößen als Zielgrößen bezeichnet. Die möglichen Wertebereiche P_i der Parameter bilden nach Nunes und Antunes [109] den Parameterraum P und eine konkrete Wertebelegung nennen sie Parameterkonfiguration P_c. Demgemäß ist der Parameterraum in Gleichung 1 und die Parameterkonfiguration in Gleichung 2 formal spezifiziert.

$$P = P_1 \times P_2 \times \ldots \times P_n \tag{1}$$

$$P_c = (p_1, p_2, \ldots, p_n), \text{ mit } p_i \in P_i \tag{2}$$

Es ist anzumerken, dass nicht nur das Simulationsmodell, sondern auch der Simulator Einflussgrößen und Ausgabegrößen besitzt. Dieser Sachverhalt wird in der M&S Literatur, wie zum Beispiel in [16, 187, 185, 166], oft vernachläßigt und wird in der Arbeit zu einem späteren Zeitpunkt erneut aufgegriffen.
Mit der Implementierung des Simulationsmodells ist die 3-phasige Modellbildung abgeschlossen und es folgt gemäß Abbildung 1 die Phase *Experimente & Analyse*, das heißt gemäß der vorangegangenen Betrachtung die Simulation.

2.1.3 Simulation als Ersatz empirischer Untersuchungen am System

Das aktive Experimentieren am System stellt nach Clymer [21] die beste Möglichkeit dar, dessen grundlegende Funktionsweise zu durchdringen. Wie eingangs erläutert, dient die Methode der M&S dazu Experimente am realen System zu reduzieren oder zu ersetzen. Demgemäß kann die Aussage von Clymer analog auf Simulationsmodelle bezogen werden. Somit ist das aktive Experimentieren mit einem Simulationsmodell die beste Möglichkeit, dessen komplexe Zusammenhänge und Sachverhalte zu verstehen. Im Weiteren werden die Begriffe Experiment und Simulationsexperiment gleichgesetzt. Die Definition des Begriffs Simulationsexperiment für diese Arbeit erfolgt durch eine Kombination der Definitionen von Philipp [120] und der VDI-Richtlinie 3633 [170]. Demgemäß ist ein *Simulationsexperiment* eine problembezogene methodisch angelegte empirische Untersuchung kausaler Zusammenhänge eines Simulationsmodells durch wiederholte Simulationsläufe mit systematischer Variation der Einflussgrößen oder der Modellstruktur. Der *Simulationslauf* bildet die Grundlage für Simulationsexperimente und entspricht genau einer Abarbeitung des Simulationsmodells durch den Simulator über eine zulässige Zeitdauer mit festgelegten Randbedingungen. Zu den Randbedingungen zählen die Festlegung der Modellstruktur und der Einflussgrößen, wobei die Modellstruktur und die Einflussgrößen sich während eines Simulationslaufes nicht ändern [178]. Nach der Durchführung der Simulationsexperimente stehen dem Anwender die Simulationsergebnisse, meist zeitbezogene Verläufe der Zielgrößen, zur Verfügung. Durch die Analyse der Simulationsergebnisse werden die kausalen Zusammenhänge abgeleitet, interpretiert und das Systemverhalten prognostiziert. Unter einer Simulationsstudie wird im Rahmen dieser Arbeit das Durchführen von einer Menge logisch geordneter Simulationsexperimente gemäß Barton [9] verstanden. Diese Interpretation grenzt sich von der allgemeinen Definition in der VDI-Richtlinie 3633 [170] ab. Das Thema Simulationsexperimente wird im nächsten Abschnitt näher beleuchtet.

2.2 Simulationsbasierte Experimente: Klassifikation und Aufbau

Als Ergebnis der Modellbildung erhält man ein abstrahiertes und auf einem Rechner ausführbares Simulationsmodell. Simulationsexperimente ermöglichen komplexe Sachverhalte und Zusammenhänge eines Simulationsmodells zu verstehen und die gewonnen Erkenntnisse können dann auf die Realität übertragen werden.

Im Rahmen der M&S existiert eine Vielzahl unterschiedlicher Experimentziele. Basierend auf den Recherchen von [79, 9, 92, 119, 80] können die wesentlichen Experimentziele allgemein wie folgt definiert werden: (i) *Validierung, Verifikation & Testing*, (ii) *Screening*, (iii) *Sensitivitätsanalyse*, (iv) *Explorative Analyse*, (v) *Optimierung*. Bei letzterer sollte nach Hagendorf [58] zwischen der klassischen *Parameteroptimierung* und einer *Struktur- und Parameteroptimierung* unterschieden werden. Streng genommen wird in der M&S jedem Experimentziel das Attribut *simulationsbasiert* (*eng. simulation-based*) vorangestellt, wie zum Beispiel *simulation-based sensitivity analysis* [27] oder *simulation-based optimization* [24]. Zur sprachlichen Vereinfachung wird im Weiteren das Attribut *simulationsbasiert* bewusst weggelassen.

Bezogen auf eine Simulationsstudie verdeutlicht Barton [9], dass die Simulationsexperimente logisch angelegt und durchgeführt werden müssen. Dafür unterteilt Barton eine

Simulationsstudie grob in drei Experimentphasen: (i) *frühzeitig*, (ii) *mittelfristig* und (iii) *langfristig*. Demnach sind auch die übergeordneten Experimentziele logisch zu ordnen. In diesem Zusammenhang zeigt die Tabelle 2 eine vereinfachte Klassifikation der übergeordneten Experimentziele, bezogen auf die Experimentphasen und den Experimentaufbau. Die Einordnung der Experimentziele bezüglich der Experimentphase basiert teilweise auf [9].

Tabelle 2: Vorschlag einer vereinfachten Klassifikation der übergeordneten Experimentziele bezogen auf den Experimentaufbau und die Experimentphase.

Experimentphase / *Experimentaufbau*	**frühzeitig**	**mittelfristig**	**langfristig**
einfach	Validierung, Verifikation & Testing; Explorative Analyse	Sensitivitätsanalyse	
komplex	Screening		Parameteroptimierung
hochkomplex			Struktur-& Parameteroptimierung

Es ist zu berücksichtigen, dass für jedes Experimentziel eine Menge unterschiedlicher numerischer Methoden existieren. Nach Lattner [90] dienen beispielsweise die numerischen Methoden *Simulated Annealing*, *Tabu Search* und *Genetic Algorithm* dem Experimentziel *Optimierung*. Im Rahmen simulationsbasierter Experimente basieren die numerischen Methoden auf der Durchführung einer Vielzahl von Simulationsläufen. Die einzelnen numerischen Methoden zum Erreichen der jeweiligen Experimentziele werden in dieser Arbeit nicht näher diskutiert, sondern es wird nur auf entsprechende Literatur verwiesen.

Zunächst werden die jeweiligen Experimentziele allgemein erläutert. Im Anschluss daran erfolgt eine Diskussion des Experimentaufbaus.

2.2.1 Beschreibung der allgemeinen Experimentziele

In diesem Unterabschnitt werden die allgemeinen Experimentziele kurz eingeführt.

Validation, Verifikation & Testing

Das Experimentziel Validierung, Verifikation und Testing (VV&T) ist bereits im Abschnitt 2.1 kurz eingeführt worden. Es bildet einen essentialen Bestandteil eines jeden M&S-Projektes. Vorangegangen wurde die VV&T jedoch ausschließlich auf das Ergebnis der Phase *Implementierung* in Abbildung 1 bezogen und damit auf das Simulationsmodell eingegrenzt. Analog Zander et al. [182] ist das Experimentziel VV&T oft weiter zufassen, beispielsweise auch für den Komponententest und Integrationstest.

Gemäß Tabelle 2 ordnet sich das Experimentziel VV&T in die Experimentphase *frühzeitig* ein. Streng genommen muss jedes Simulationsmodell als allererstes der VV&T zur Ermittlung der vollständigen Korrektheit bezogen auf die zu lösende Problemstellung unterzogen werden [134]. Damit sollen Fehlentscheidungen und Fehlinterpretationen vermieden werden. Vor diesem Hintergrund sind alle weiteren Experimentziele erst nach einer erfolgreichen VV&T anzuordnen. Jedoch merken Rabe et al. [124] an, dass grundsätzlich eine *vollständige* Korrektheit eines Simulationsmodells nicht nachgewiesen werden kann.

Wie bereits in Abschnitt 2.1 diskutiert wird im Rahmen dieser Arbeit die VV&T Thematik nicht eingehender betrachtet, da sie quasi als separates Gebiet umfassend untersucht wird, wie zum Beispiel in [5, 134, 8, 182, 169, 124, 119].

Explorative Analyse

Soykan [151] beschreibt eine explorative Analyse als die Durchführung von explorativen Simulationsexperimenten zur Ermittlung von ausschlaggebenden und signifikanten Parametern eines Simulationsmodells. Aus der Sicht des Autors ist diese Aussage zu kurz gefasst und entspricht eher den Experimentzielen *Screening* und *Sensitivitätsanalyse*, welche auf den nachfolgenden Seiten näher beschrieben werden. Daher wird in dieser Arbeit die explorative Analyse als eine grundlegende Untersuchung des Simulationsmodells mit dem Ziel der Förderung des Verständnisses oder des Erkenntnisgewinns über das Simulationsmodell verstanden. Denn das ist essentiell für die Durchführung von komplexen und hochkomplexen Simulationsexperimenten nach Tabelle 2.

Wie das Attribut *explorativ* verdeutlicht, steht hier das *Erforschen* des Simulationsmodells im Vordergrund. Ein solches Experimentziel wird dann essentiell, wenn sich zum Beispiel ein Anwender in ein Simulationsmodell Dritter einarbeiten muss. Vor diesem Hintergrund zählt eine einfache Parameterstudie, also das Durchführen einer Menge von systematischen oder unsystematischen Simulationsläufen, am Simulationsmodell zu einer explorativen Analyse. Leye [92] betrachtet die Berechnung einfacher prädiktiver Ergebnisse als explorative Analyse. Günther und Veltner [55] zählen im erweiterten Sinne die statistische Auswertung von Simulationsergebnissen für den Erkenntnisgewinn zu einer explorativen Analyse. Weiterhin können die von Kelton und Barton [76] formulierten Fragen:

1. Welche Konfigurationen der Einflussgrößen sollen ausgeführt werden?
2. Wie lange soll ein Simulationslauf dauern?
3. Wie viele Simulationsläufe sollen durchgeführt werden?
4. Wie sind die Simulationsergebnisse zu interpretieren und zu analysieren?
5. Was ist der effizienteste weg um Simulationsläufe auszuführen?

unter Verwendung der explorativen Analyse untersucht werden. Insofern kann diese der *frühzeitigen* sowie *mittelfristigen* Experimentphase zugeordnet werden.
Abschließend soll erwähnt werden, dass keine formale Vorgehensweise zur Durchführung von explorativen Analysen existiert und diese somit anwendungsspezifisch, basierend auf der Intuition und Erfahrung des Anwenders, durchzuführen sind.

Screening

Ab einer bestimmten Komplexität des Simulationsmodells stellt sich beim Experimentieren schnell die Frage: „*Welche der unzähligen Einflussgrößen haben einen signifikanten Einfluss auf die Ausgabegrößen?*“. Kleijnen [79] bringt in diesem Zusammenhang die 80-zu-20-Regel von Vilfredo Pareto (1848-1923) ins Spiel. Vor diesem Hintergrund angewendet besagt die Regel, dass lediglich 20% der Einflussgrößen einen Einfluss von 80% auf die Ausgangsgrößen eines Simulationsmodells besitzen. Genau hier setzt das *Screening* an. Es beschäftigt sich mit der Identifikation der Menge der signifikanten Einflussgrößen aus der Menge der gesamten Einflussgrößen eines Simulationsmodells in der *frühzeitigen* Experimentphase. Beim Screening werden die Einflussgrößen eines Simulationsmodells

Faktoren und der zugehörige Parameterraum als Faktorraum bezeichnet. Das Experimentziel von Screening ist die Ermittlung *qualitativer* Sensitivitätsmaße der einzelnen Faktoren [15]. Anhand der Sensitivitätsmaße kann jeder Faktor in *signifikant* oder *nicht-signifikant* eingeteilt werden. Unterschiedliche numerische Methoden zum Screening werden beispielsweise in [79, 15, 30, 26, 13] präsentiert und diskutiert. Diese besitzen meist ein sehr gutes Laufzeitverhalten und sind somit auf Simulationsmodelle mit mehreren hundert Faktoren anwendbar.

Im Anschluss an das Screening können nicht-signifikante Faktoren auf einen beliebigen Wert des zugehörigen Wertebereiches fixiert werden und brauchen nicht weiter betrachtet werden. Demnach wird der Parameterraum (P) des Simulationsmodells gemäß Gleichung 1 auf den Parameterraum der signifikanten Faktoren ($P_{SC,sig}$) eingeschränkt. Diesen Sachverhalt zeigt Gleichung 3.

$$P_{SC,sig} \subseteq P \tag{3}$$

Auf diese Weise reduziert sich die Anzahl der variierbaren Einflussgrößen, was die Transparenz des Simulationsmodells erhöht. Leider können keine Schlüsse gezogen werden, wie stark die Faktoren der Gruppe *signifikant* die Ausgabegrößen wirklich beeinflussen. Genau hier setzt die Sensitivitätsanalyse für eine detaillierte Faktoranalyse an.

Sensitivitätsanalyse

Sensitivitätsanalysen (SA) besitzen grundlegend den gleichen Fokus wie Screening [131] und werden gemäß Tabelle 2 in der *mittelfristigen* Experimentphase eingesetzt. Jedoch ist das Ziel der Sensitivitätsanalyse die Berechnung *quantitativer* Sensitivitätsmaße der einzelnen Faktoren eines Simulationsmodells. Hervorzuheben sind an dieser Stelle vor allem die varianzbasierten Sensitivitätsanalysen [61, 132]. SA dieser Kategorie generieren für jeden Faktor eine Stichprobe an Hand von Verteilungsfunktionen unter Beachtung der zulässigen Wertebereiche der Faktoren. Dabei kann jedem Faktor eine andere Verteilungsfunktion zugeordnet werden. Aus den Simulationsergebnissen können durch eine Varianzdekomposition Sensitivitätsmaße als prozentuale Größen bestimmt werden und somit der prozentuale Einfluss einzelner Faktoren auf die Ausgabegrößen des Simulationsmodells beschrieben werden [132, 20].

Die Sensitivitätsmaße sind total geordnet und dadurch untereinander vergleichbar. Faktoren mit geringen Sensitivitätsmaßen können aus der Liste der signifikanten Faktoren entfernt werden und analog dem Screening auf einen beliebigen Wert des zugehörigen Wertebereichs fixiert werden. Problematisch ist dabei die Bestimmung der Grenze ab wann ein Faktor als signifikant oder nicht-signifikat gilt. Besitzt ein Faktor beispielsweise nach der Durchführung einer SA 6% Einfluss auf eine Ausgabegröße, so ist die Einteilung des Faktors in signifikant oder nicht-signifikant schwierig.
Durch eine quantitative Sensitivitätsanalyse kann der Parameterraum $P_{SC,sig}$, welcher duch das Screening im ersten Schritt eingeschränkt wurde, weiter zu $P_{SA,sig}$ eingeengt werden. Diese Tatsache ist in Gleichung 4 gezeigt.

$$P_{SA,sig} \subseteq P_{SC,sig} \subseteq P \tag{4}$$

Es sein angemerkt, dass das Screening nicht in jedem Fall vor einer Sensitivitätsanalyse erfolgen muss.

Eine detaillierte Sensitivitätsanalyse erfordert eine Vielzahl von Simulationsläufen. Die Anzahl der Läufe hängt nach Ekström und Broed [35] direkt von der Menge zu untersuchender Faktoren ab und kann nach Meinung von Iooss und Lemaître [70] sehr schnell bis zu einer Größenordnung von 1000 Simulationsläufen oder darüber hinaus anwachsen. Analog zum Screening existierten auch zur Durchführung einer SA unterschiedliche numerische Methoden, wie zum Beispiel in [70, 132, 13, 22, 20] gezeigt.

Optimierung

In vielen Teilbereichen der Wissenschaft spielt die Optimierung eine wesentliche Rolle. Allerdings wird oft in den Ingenieurswissenschaften und insbesondere im Anwendungsfeld Produktion und Logistik (P&L) der Begriff Optimierung unterschiedlich verwendet. Zum einen wird die Optimierung oft mit der Simulation gleichgesetzt und zum anderen wird der Begriff Optimierung übergeordnet für jegliche Art der Verbesserung eines Systems genutzt [37]. März et al. [99] charakterisieren eine Optimierung für das Anwendungsfeld P&L als die Suche nach einem Optimum bei gegebenen Voraus- und Zielsetzungen. Im Rahmen der vorliegenden Arbeit soll unter einer Optimierung eine systematisch angelegte Untersuchung unter Verwendung einer numerischen Optimierungsmethode zur Ermittlung eines Minimums oder Maximums einer Zielfunktion verstanden werden. Nach Papageorgiou et al. [111] stellt eine Zielfunktion einen mathematischen Zusammenhang zwischen dem *Gegebenen* und *Gesuchten* eines Optimierungsproblems dar und bildet nach Weicker [175] den wichtigsten Bestandteil eines Optimierungsproblems. In Anlehnung an Papageorgiou et al. [111] zeigt Gleichung 5 die allgemeine mathematische Definition eines Optimierungsproblems und Gleichung 6 zeigt die Überführung eines Minimierungsproblems mittels Negation in ein Maximierungsproblem.

$$\min_{\vartheta \in \Theta} f(\vartheta), \text{ wobei } \Theta = \{\vartheta \mid c(\vartheta) = 0; h(\vartheta) \leq 0\} \tag{5}$$

$$\max_{\vartheta \in \Theta} f(\vartheta) = -\min_{\vartheta \in \Theta}(-f(\vartheta)) \tag{6}$$

Demnach wird die optimale Konfiguration der Einflussgrößen ϑ_{opt} aus einem zugehörigen Suchraum Θ, bei der die Zielfunktion f minimal oder maximal ist, gesucht. Ein Suchraum entspricht nach Kruse et al. [85] der Menge der potentiellen Lösungen einer Zielfunktion. Gleichung 7 verdeutlicht dies. Durch das vorgelangerte Screening und einer Sensitivitätsanalyse können nicht-signifikante Einflussgrößen eliminiert und daurch die Dimension des Suchraums eines Optimierungsproblems erheblich reduziert werden.

$$\vartheta_{opt} \in \Theta \subseteq P_{SA,sig} \subseteq P_{SC,sig} \subseteq P \tag{7}$$

Die Vektorfunktion c in Gleichung 5 kennzeichnet die Gleichungsnebenbedingungen und h entspricht einer Ungleichheitsnebenbedingung. Eine global optimale Lösung erfordert das Finden der Parameterkonfiguration ϑ_{opt}, bei welcher die Zielfunktion $f(\vartheta_{opt})$ den minimalen beziehungsweise maximalen Wert besitzt. Allerdings ist das Finden der global optimalen Lösung bei numerischen Ansätzen nicht immer gegeben, sodass man sich auch oft mit suboptimalen Lösungen, also dem Finden lokaler Minima oder Maxima, zufriedengeben muss.

Weiterhin muss nach Weicker [175] der Erstellung einer Zielfunktion hinreichende Aufmerksamkeit gewidmet werden. Denn eine falsch gewählte Zielfunktion führt automatisch zu falschen, unzureichenden oder keinen Ergebnissen.

Hagendorf [58] führt eine erweiterte Methode zur Struktur- und Parameteroptimierung ein. Dabei zeigt Hagendorf, wie neben den Einflussgrößen eines Simulationsmodells auch dessen Struktur innerhalb eines Optimierungsproblems mitbetrachtet werden kann. Eine eingehende Betrachtung dieser Methode erfolgt im nächsten Unterabschnitt.

Talbi [157] unterstreicht den immensen Popularitätsgewinn der metaheuristischen Optimierungsmethoden in den letzten 20 Jahren und prognostiziert eine weitere Zunahme der Popularität. Gründe dafür sind die Effizienz und Effektivität mit der Optimierungsprobleme selbst mit großen Suchräumen gelöst werden können. Unterschiedliche numerische Optimierungsmethoden werden zum Beispiel in [111, 16, 38, 62, 11, 155] präsentiert.

Nach der Einführung der übergeordneten Experimentziele wird nachfolgend auf den Experimentaufbau gemäß den Kategorien in Tabelle 2 eingegangen.

2.2.2 Aufbau von simulationsbasierten Experimenten

In diesem Abschnitt wird auf den prinzipiellen Experimentaufbau gemäß der Einteilung in (i) *einfach*, (ii) *komplex* und (iii) *hochkomplex* in Tabelle 2 eingegangen.

Einfache Simulationsexperimente

Der Aufbau einfacher Experimente ist in Abbildung 4 dargestellt. Sie umfassen einen Simulator, das Simulationsmodell und eine Experimentsteuerung.

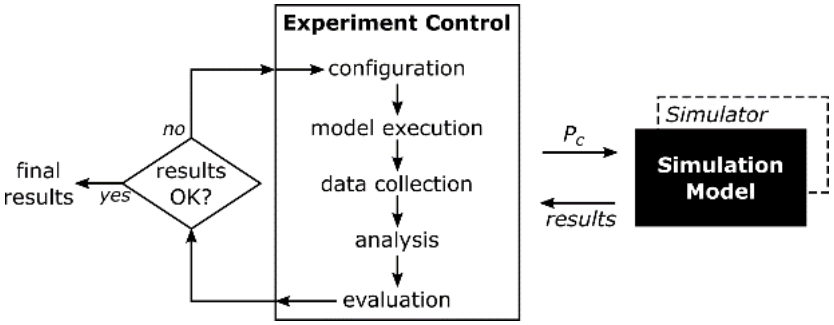


Abbildung 4: Aufbau eines einfachen Simulationsexperiments.

Wie in Abschnitt 2.1 beschrieben, entspricht das modular-hierarchische Simulationsmodell beim Experimentieren einer Black-Box. Insofern sind nur die Einflussgrößen und die Ausgabegrößen von wesentlichem Interesse. Die interne Codierung der Dynamik und der Struktur des Simulationsmodells sind für das Experiment selbst *uninteressant*. Gemäß Heitmann [64] ist eine Experimentsteuerung notwendig, um Experimente, also die Durchführung eines oder mehrere Simulationsläufe, automatisiert ablaufen zu lassen. Nach Kolitz [84] muss die Experimentsteuerung eine konkrete Ablaufsteuerung des durchzuführenden Experimentes codieren. Die eindeutige Trennung zwischen Experimentsteuerung und Simulationsmodell trägt zu der von vielen Autoren, wie zum Beispiel in [187, 102, 185, 168, 167], geforderten Trennung von Modell- und Experimentbeschreibung bei. Auf diese Weise kann die Flexibilität, Effektivität, Wiederverwendbarkeit, Wartbarkeit und die

Erweiterbarkeit sowohl des Simulationsmodells als auch der Experimentsteuerung verbessert werden.

Zur Diskussion der Experimentsteuerung werden im Weiteren die von Leye [92] identifizierten charakteristischen Aufgaben von Simulationsexperimenten aufgegriffen. Wie in Abbildung 4 zu sehen, sind das die: (i) *Konfiguration*, (ii) *Modellausführung*, (iii) *Datensammlung*, (iv) *Analyse* sowie (v) *Bewertung*.

Die Experimentsteuerung wird durch den Anwender erstellt und beschreibt detailliert die logisch aufeinander aufbauenden Aufgaben zur Untersuchung einer oder mehrerer Problemstellungen. Sie verknüpft die Aufgaben direkt mit der Experimentausführung. Demnach kann die Experimentsteuerung mit einer ausführbaren Experimentspezifikation gleichgesetzt werden. Im Weiteren werden die einzelnen Aufgaben der Experimentsteuerung basierend auf Leye [92] beleuchtet. Die *Konfiguration* beschreibt die Auswahl der Parameterkonfiguration $P_c \in P$, gemäß Gleichung 1 und 2, und damit die Konfiguration der Einflussgrößen des Simulationsmodells, bezogen auf eine zu untersuchende Problemstellung. Die *Modellausführung* beschäftigt sich mit der Abarbeitung des Simulationsmodells durch einen Simulator und somit mit der Ausführung von einem oder mehreren Simulationsläufen zur Berechnung der erforderlichen Simulationsergebnisse. Hierzu erfolgt die Festlegung eines Simulators zur Abarbeitung des Simulationsmodells. Wie bereits im Unterabschnitt 2.1.2 diskutiert, besitzt auch der Simulator Einflussgrößen, die im Rahmen der Modellausführung festzulegen sind. Beispielsweise kann die Zeitdauer ($t_{start} \rightarrow t_{final}$) als eine wesentliche Einflussgröße des Simulators angesehen werden. Als nächstes folgt die Aufgabe der *Datensammlung*. Sie spezifiziert die Extraktion und Aufbereitung relevanter Simulationsergebnisse nach einem Simulationslauf. Die *Datenanalyse* umfasst die Verdichtung oder Ableitung wesentlicher Informationen zu den Zielgrößen. Beispielsweise zählen im Anwendungsbereich Produktion und Logistik (i) die maximale Maschinenauslastung, (ii) die maximale Pufferkapazität und (iii) die Durchlaufzeit zu den wesentlichen Zielgrößen. Die Aufgabe *Bewertung* prüft die durch die *Datenanalyse* gewonnen Ergebnisse. Es wird zum Beispiel geprüft, ob alle notwendigen Randbedingungen und/oder Toleranzen eingehalten werden und es kann eine erste einfache Plausibilitätsprüfung der Ergebnisse erfolgen. Nach Abschluss eines Experimentes werden alle Ergebnisse dem Anwender zur endgültigen Bewertung zur Verfügung gestellt. Fällt das abschließende Urteil *positiv* aus, so ist der aktuelle Experimentprozess beendet. Im anderen Fall wird der Experimentprozess mit der Aufgabe *Konfiguration* fortgesetzt.

Für eine automatisierte Ausführung von Simulationsexperimenten muss die Experimentsteuerung, also alle genannten Aufgaben, unter Verwendung einer allgemeinen Programmiersprache, einer Berechnungsumgebung, wie zum Beispiel MATLAB [158] und Scilab [150], oder einer DSL codiert werden. Unterschiedliche DSL zur Spezifikation von Simulationsexperimenten werden von Schützel et al. [147] präsentiert und diskutiert.
Der einfache Experimentaufbau wird gemäß Tabelle 2 vornehmlich bei der explorativen Analyse, bei der VV&T oder bei der Sensitivitätsanalyse eingesetzt. Bei der explorativen Analyse können beispielsweise zur Erforschung des Simulationsmodells unterschiedliche ausführbare Experimentspezifikationen formuliert und automatisiert ausgeführt werden. Bei der VV&T können beispielsweise unterschiedliche Testfälle mit den zugehörigen Testdaten,

den zu erwarteten Ergebnissen und der Testbewertung spezifiziert werden. Bei der Sensitivitätsanalyse kann der einfache Experimentaufbau speziell bei den graphischen Methoden [132, 13, 22] zum Einsatz kommen.

Es ist ersichtlich, dass ein solcher Experimentaufbau nur zur Lösung einfacher Problemstellungen geeignet ist. Bei komplexen Problemstellungen muss dieser entsprechend erweitert werden.

Komplexe Simulationsexperimente

Im Gegensatz zu den einfachen Experimenten besitzen *komplexe* Experimente eine *Experimentmethode*, die automatisiert Simulationsläufe ausführt, die Simulationsergebnisse analysiert, speichert und gegebenenfalls neue Simulationsläufe konfiguriert und ausführt. Allgemein stellen komplexe Simulationsexperimente einen Feedback-Prozess dar. Sie variieren automatisiert Einflussgrößen eines Simulationsmodells und führen Simulationsläufe zur Erreichung eines Experimentzieles aus. Unter einer Experimentmethode wird in dieser Arbeit eine programmtechnisch umgesetzte numerische Methode zum Erreichen eines übergeordneten Experimentziels gemäß Tabelle 2 verstanden. Der prinzipielle Aufbau komplexer Experimente soll am Beispiel einer Optimierung diskutiert werden. Hierzu zeigt Abbildung 5 stellvertretend zwei in der Literatur anerkannte Strukturierungsmöglichkeiten einer Optimierung. Zunächst werden die beiden Ansätze separat vorgestellt und im Anschluss gegenübergestellt.

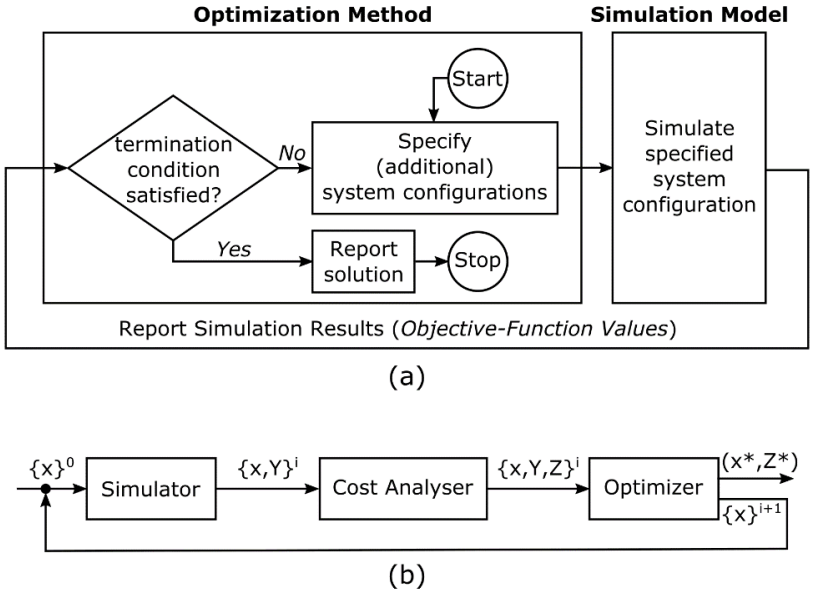


Abbildung 5: Strukturierung einer Optimierung nach: (a) Law und Kelton [91] und (b) Han et al. [60].

Ansatz nach Law und Kelton [91]

Der Ansatz von Law und Kelton [91] besteht aus den miteinander verknüpften Komponenten *Optimierungsmethode* und *Simulationsmodell*. Die Optimierungsmethode stellt die Experimentmethode dar und codiert den charakteristischen Suchalgorithmus einschließlich den Abbruchkriterien. Am Anfang berechnet die Experimentmethode eine Konfiguration der Einflussgrößen $\vartheta_i \in \Theta$ des Simulationsmodells, in Abbildung 5a *system configuration*

genannt. Die Komponente Simulationsmodell führt einen Simulationslauf durch und liefert im Anschluss die Simulationsergebnisse sowie den zu minimierenden Zielfunktionswert $f(\vartheta_i)$ an die Experimentmethode zurück. Somit ist die Zielfunktion ein integraler Bestandteil des Simulationsmodells, was auch dem Vorgehen von Deckert [28] sowie Fu [46] entspricht. Abschließend wird geprüft, ob ein Abbruchkriterium erfüllt ist, um dann die optimale Konfiguration der Einflussgrößen auszugeben und die Optimierung abzubrechen oder im anderen Fall auf Basis des Zielfunktionswertes eine neue Konfiguration der Einflussgrößen zu berechnen und die Optimierungsschleife erneut zu durchlaufen.

Ansatz nach Han et al. [60]

In Abbildung 5b ist die Struktur einer Optimierung nach Han et al. [60] schematisch dargestellt. Demnach besteht die Optimierung aus den Komponenten *Simulator*, *Cost Analyser* und *Optimizer*. Letzteres entspricht im aktuellen Sprachgebrauch einer Experimentmethode. Weiterhin entspricht x dem $\vartheta_i \in \Theta$ gemäß Abschnitt 2.2.1. Y repräsentiert die Simulationsergebnisse und Z entspricht dem Zielfunktionswert $f(\vartheta_i)$. Im Unterschied zum Ansatz nach Law und Kelton wird die Zielfunktion nicht als Bestandteil des Simulationsmodells angesehen, sondern separat im *Cost Analyser* spezifiziert. Ein solches Vorgehen fordern auch die Autoren Talbi [157] und Gehlsen [49]. Demnach unterstützt die Trennung zwischen Zielfunktion und Simulationsmodell effiziente und flexible Wiederverwendbarkeit sowie Adaption für weitere M&S-Projekte.

Zu Beginn wird der Simulator mit dem Startwert $\{x\}^0$ parametriert, ein Simulationslauf durchgeführt und die Simulationsergebnisse zusammen mit dem Startwert $\{x,Y\}^i$ an die Komponente *Cost Analysis* übermittelt. Dieser berechnet auf Basis von $\{x,Y\}^i$ den Zielfunktionswert Z und übermittelt die Menge $\{x,Y,Z\}^i$ an die Experimentmethode, den *Optimizer*. Letzterer wird von Han et al. [60] nicht detailliert behandelt. Es wird davon ausgegangen, dass der *Optimizer* analog zur Optimierungsmethode in Abbildung 5a aufgebaut. Das heißt, die Experimentmethode *Optimizer* prüft das Abbruchkriterium, bewertet das aktuelle Ergebnis und berechnet die neue Konfiguration der Einflussgrößen $\{x\}^{i+1}$ oder bricht die Experimentschleife ab und gibt die gefundene Lösung in Form von (x^*,Z^*) aus.

Diskussion der Ansätze

Auf den ersten Blick erscheinen beide Ansätze ähnlich. Nachfolgend werden die Unterschiede herausgearbeitet und es wird auf einige konzeptionelle Unklarheiten eingegangen.
Im Ansatz von Law und Kelton wird der Simulator nicht explizit angegeben. Damit stellt sich die Frage, wer das Simulationsmodell abarbeitet. Genauso ist unklar, wie die Zielfunktion im Simulationsmodell zu codieren ist. Der Ansatz von Han et al. trennt nicht explizit zwischen Simulator und Simulationsmodell. Damit ist unklar worauf sich die Einflussgrößen $\{x\}^{i+1}$, die durch die Experimentmethode variiert werden, beziehen. Law und Kelton weisen explizit daraufhin, dass die Experimentmethode eine Vielzahl einstellbarer Optionen besitzt, wie zum Beispiel Toleranzen für das Abbruchkriterium, Parametrierung der Suchstrategie und Wahl des Startwertes, die sich auf das Gesamtergebnis auswirken können [91]. Dennoch ist in beiden Ansätzen in Abbildung 5 jegliche Parametrierung der Experimentmethode weggelassen. Abschließend bleibt auch unklar, wie die Simulationsläufe konkret auszuführen sind. Im vorangegangenen Abschnitt *Einfache Simulationsexperimente* wurde gezeigt, dass die Durchführung von Simulationsläufen nicht trivial ist.

Die beschriebenen und zum Teil weiteren konzeptuellen Unklarheiten finden sich leider in vielen Arbeiten, wie zum Beispiel auch in Soykan [151], Gehlsen [49], Fink [41], Syberfeldt et al. [156] sowie März und Krug [100], wieder. Aus diesem Grund ist aus Sicht des Autors die Entwicklung einer konzeptuell sauberen Strukturierung komplexer Simulationsexperimente zwingend notwendig.

Bei den hier diskutierten komplexen Simulationsexperimenten wurde von einem Simulationsmodell mit zeitinvarianter Modellstruktur ausgegangen. Das Experiment variiert nur die Einflussgrößen eines Simulationsmodells. Somit bildet die Struktur des Simulationsmodells keinen Bestandteil des Experimentprozesses. Genau an diesem Punkt setzen die hochkomplexen Simulationsexperimente nach Tabelle 2 an.

Hochkomplexe Simulationsexperimente

Hochkomplexe Simulationsexperimente beziehen neben der automatisierten Variation der Einflussgrößen auch die automatisierte Variation der Struktur eines Simulationsmodells mit ein. Die Grundprinzipien *hochkomplexer* Simulationsexperimente werden nachfolgend am Beispiel der von Hagendorf [58] eingeführten simulationsbasierten Struktur- und Parameteroptimierung erläutert. Den prinzipiellen Aufbau und Ablauf zeigt Abbildung 6.

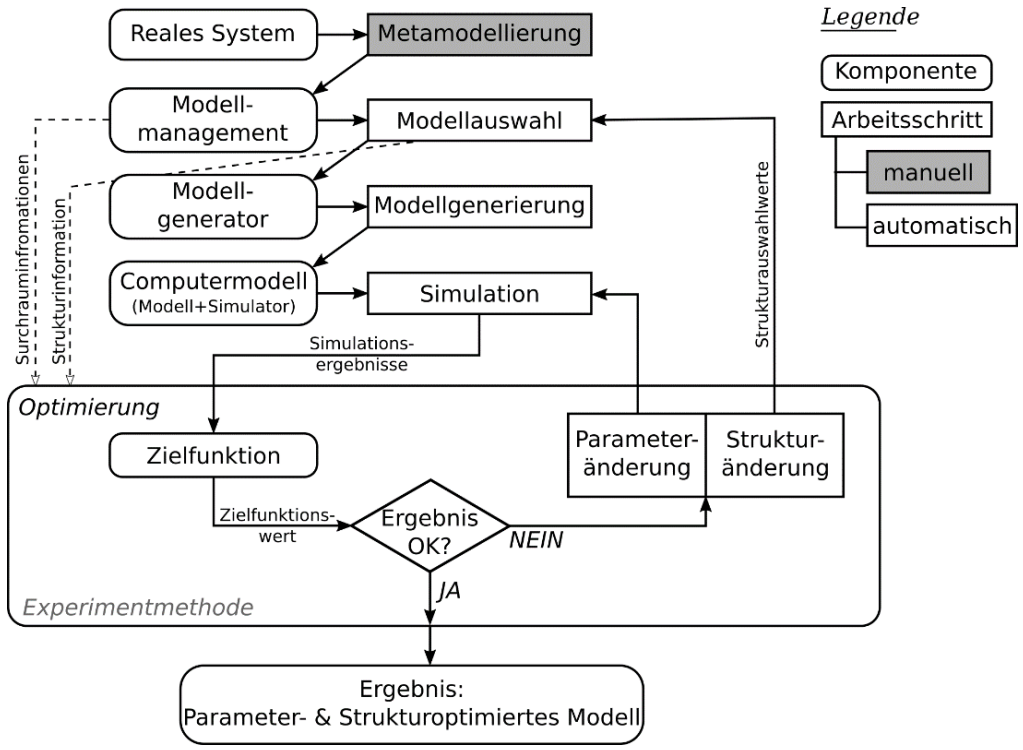


Abbildung 6: Aufbau und Ablauf eines hochkomplexen Experiments am Beispiel einer kombinierten Struktur- und Parameteroptimierung nach Hagendorf [58].

Die Idee basiert auf der Kopplung einer Experimentmethode, hier eine Optimierungsmethode, mit einer Modellgenerierungseinheit zur automatischen Generierung ausführbarer Simulationsmodelle. Dabei fordert Hagendorf die Metamodellierung eines Systems. Es wird also nicht mehr ein Simulationsmodell mit einer festen Struktur modelliert, sondern ein *Metamodell*. Hagendorf definiert ein Metamodell als ein Modell, welches das Verhalten und die Struktur unterschiedlicher Modelle eines Systems spezifiziert [58]. Die Komponente

Modellmanagement verwaltet das Metamodell und stellt die notwendigen Methoden zur Auswahl eines konkreten Modells zur Verfügung. Weiterhin stellt die Komponente *Modellmanagement* die Suchrauminformationen der Optimierungsmethode zur Verfügung. Der Suchraum Θ besteht jetzt nach Gleichung 8 aus dem Kreuzprodukt des klassischen Suchraums Θ_P eines Simulationsmodells mit der Menge aller möglichen Strukturen eines Simulationsmodell Θ_S.

$$\Theta = \Theta_P \times \Theta_S \tag{8}$$

Durch den Arbeitsschritt *Modellauswahl* wird genau festgelegt, welches Simulationsmodell bezüglich Verhalten und Struktur ausgewählt wird. Unter Verwendung eines *Modellgenerators* und des Arbeitsschrittes *Modellgenerierung* wird ein *Computermodell* generiert. Hagendorf definiert den Begriff als eine Komposition eines Simulators mit einem Modell. Durch die *Modellgenerierung* wird ein Simulationsmodell generiert, mit dem zugehörigen Simulator verschaltet und konfiguriert. Im Arbeitsschritt *Simulation* wird ein Simulationslauf ausgeführt. Danach werden die Simulationsergebnisse durch die Zielfunktion bewertet und die Optimierungsmethode prüft, ob ein Abbruchkriterium erreicht wurde oder nicht. Im letzten Fall erfolgt eine Änderung der Parameter und gegebenenfalls der Struktur des Simulationsmodells. Bei der Änderung der Struktur werden dem Arbeitsschritt *Modellauswahl* die notwenigen Strukturauswahlwerte übermittelt. Es kann ein neues Simulationsmodell generiert werden und es werden die beschriebenen Arbeitsschritte erneut durchlaufen. Am Ende steht dem Anwender ein parameter- und strukturoptimiertes Simulationsmodell zur Verfügung.

Natürlich erfordert die Spezifikation eines hochkomplexen Experiments einen erheblichen Zeitaufwand. Jedoch besitzen hoch komplexe Experimente durch die automatisierte Generierung von Simulationsmodellen und die automatisierte Ausführung von Simulationsläufen ein umfangreiches Experimentierpotenzial.

2.2.3 Zwischenfazit

Wie gezeigt, gliedern sich die übergeordneten Experimentziele in die vier Klassen (i) *Validierung*, *Verifikation* und *Testing*, (ii) *Screening*, (iii) *Sensitivitätsanalyse* und (iv) *Optimierung*. Wobei in dieser Arbeit bei der Optimierung zwischen reiner Parameteroptimierung sowie kombinierter Struktur- und Parameteroptimierung unterschieden wird. Anhand der Literatur ist auch gezeigt worden, mit Ausnahme des Experimentziels der VV&T, dass für die unterschiedlichen Kategorien der Experimentziele eine Menge unterschiedlicher Experimentmethoden existieren. Im klassischen Sinne muss sich der Anwender beim Durchführen von Simulationsexperimenten immer für eine oder mehrere konkrete Experimentmethoden entscheiden und die gewünschten Experimente gemäß den Experimentzielen spezifizieren. Natürlich stellt sich hier die Frage, welche Experimentmethode für ein entsprechendes Experimentziel die *richtige* ist, um am schnellsten ein akzeptables Ergebnis zu erreichen. Weiterhin besitzen die Experimentmethoden eine Vielzahl an Parametrierungsmöglichkeiten. Bei der Optimierung besitzt zum Beispiel die Experimentmethode *Genetischer Algorithmus* einstellbare Parameter wie: Populationsgröße, Anzahl der Generationen, Typ der Kreuzung, Kreuzungsrate, Mutationstyp oder

Mutationsrate [155]. Damit hängt der Erfolg einer Optimierung auch direkt von den einstellbaren Parametern ab. Damit steigt die Variantenvielfalt sowie der damit einhergehende *Verwirrungsgrad* mit dem die Anwender bei der Durchführung von M&S-Projekten konfrontiert werden. Daraus folgt eindeutig die Notwendigkeit eines umfassenden Variantenmanagements zur Beherrschung der Vielfalt und damit zur Reduktion des Verwirrungsgrades.

Zwar beschäftigen sich einige Autoren, wie zum Beispiel Dupuy et al. [30], Ekström und Broed [35], Figueira und Almada-Lobo [38], Talbi [157], Sumathi und Surekha [155], Frey und Patil [44], Kleppmann [81] sowie Lobo et al. [95], teilweise mit der Entwicklung von Methoden, um dem steigenden Verrwirrungsgrad entgegen zu wirken. Sie liefern jedoch nur problemspezifische Ansätze und keine generell anwendbare Methodik.

Für das übergeordnete Experimentziel der Optimierung empfiehlt Talbi eine heuristische Herangehensweise [157]. Wurde beispielsweise eine Problemstellung A mit einer Experimentmethode und zugehöriger Parametrierung gelöst, so kann die daraus gewonnene Erfahrung durch Analogiebetrachtung zur Lösung ähnlicher Problemstellungen herangezogen werden. Ist ein solches Vorgehen nicht möglich, weil eine Problemstellung zum Beispiel noch nicht bearbeitet wurde, so verweisen Talbi [157] sowie Fink und Rothlauf [39] auf ein intensives Literaturstudium. Es ist natürlich fragwürdig, inwiefern ein intensives Literaturstudium zur Beantwortung der Fragen in der verfügbaren Bearbeitungszeit eines M&S Projektes möglich ist und ob es entsprechende Lösungsansätze liefert. Natürlich ist eine solche heuristische Herangehensweise auch auf die anderen übergeordneten Experimentziele übertragbar.

Mit dem Projekt *AssistSim* versuchen Lattner et al. [89] diesem Problem in der M&S im Anwendungsbereich der Produktion und Logistik entgegen zu wirken. Das übergeordnete Ziel ist die Entwicklung einer Entscheidungsunterstützungssoftware, die eine standardisierte, systematische sowie (teil-)automatische Experimentplanung und – durchführung ermöglicht. Die methodische Basis der Experimentplanung beruht auf Checklisten, die Expertenwissen unterschiedlicher M&S Projekte codieren [12]. Auf Grundlage der Checklisten werden bei der Experimentdurchführung die Anzahl und Dauer einzelner Simulationsläufe und die unter statistischen Gesichtspunkten notwendige Anzahl Replikationen bestimmt. Somit grenzt sich dieses Projekt ausschließlich auf die Experimentplanung und – durchführung von Simulationsläufen ein und bringt keine allgemeine Lösung.

Leye begegnet der Problemstellung mit der von ihm entwickelten *GUIding Simulation Experiments* (GUISE) Methodik [92] für die wissenschaftliche M&S-Umgebung JAMES II [66]. Die aufgabenorientierte Methodik ermöglicht es alle Simulationsexperimente in Teilaufgaben zu zerlegen und jeder Teilaufgabe eine konkrete Methode zu zuordnen. Weiterhin wurde ein synthetischer Problemlösungsmechanismus umgesetzt, welcher auf Basis der zu lösenden Problemstellung den einzelnen Teilaufgaben automatisiert entsprechende Methoden zuordnet. Damit erfolgt eine Unterstützung des Anwenders bei der Ermittlung einer geeigneten Lösung für die vorliegende Problemstellung. Jedoch bleibt die oft umfangreiche Parametrierung der jeweiligen Methoden dem Anwender überlassen. Somit liefert die von Leye präsentierte Lösung nur Unterstützung auf der Seite der Methodenauswahl. Weiterhin wird vom Autor der vorliegenden Arbeit die Akzeptanz des abstrakten aufgabenorientierten Ansatzes im Umfeld der Ingenieurwissenschaften als eher kritisch eingeschätzt.

Gemäß Lattner et al. [89] erfolgt heute die Planung, Ausführung und Bewertung von Simulationsexperimenten hauptsächlich manuell. Die Variantenvielfalt wird händisch durch möglichst *systematisches Ausprobieren* gelöst. Es wird versucht eine Vielzahl unterschiedlicher Experimente im Rahmen des verfügbaren Zeitbudgets manuell programmtechnisch umzusetzen und auszuwerten. Das heißt, eine Menge verschiedener Simulationsexperimente mit unterschiedlichen Experimentmethoden sowie Parametrierungen müssen am Simulationsmodell ausgeführt werden. Im Fall der diskutierten komplexen Experimente sind die Simulationsexperimente nicht nur an einem Simulationsmodell, sondern an mehreren Simulationsmodellen durchzuführen. Vor diesem Hintergrund ergeben sich für simulationsbasierte Experimente Fragenstellungen, wie sie analog aus anderen ingenieurtechnischen Bereichen bekannt sind und allgemein mit dem Begriff Variantenmanagement oder Variability-Management zusammengefasst werden. Zur Problematik des Variantenmanagements existieren in der M&S bereits Lösungsansätze. Diese beschränken sich jedoch auf die Modellbildung und damit im weitesten Sinne auf das Management von Simulationsmodellen. Sie berücksichtigen in keiner Weise das Experimentieren mit Simulationsmodellen. Im nachfolgenden Abschnitt wird der Stand der Forschung zum Variantenmanagement in der M&S analysiert und es werden ausgewählte Ansätze diskutiert.

2.3 Variantenmanagement in der Modellbildung und Simulation

Wie bereits im Abschnitt 2.1 diskutiert, ist die M&S heute Teil des modernen Entwicklungsprozesses in den Ingenieurwissenschaften. Der Einzug der Variantenvielfalt in die M&S erfolgte zum großen Teil über die Automobilindustrie [56]. So führten zum Beispiel die sukzessive Steigerung der Fahrzeug- und Funktionsvielfalt sowie die wachsende Bedeutung der Software in der Autoindustrie zu vielfältigen Fahrzeugtypen [68, 67, 71]. Dabei werden Fahrzeuge weniger von Grund auf neu entwickelt, sondern bestehende Komponenten mit ähnlicher Funktionalität adaptiert und/oder systematisch wiederverwendet. Das heißt, das neue Fahrzeug ist *nur* eine andere Ausprägung eines anderen Fahrzeugs. Solche Ansätze werden zum Beispiel als Baukasten- oder Plattformstrategie [136] oder Modulstrategie [71] bezeichnet. Im Rahmen der Konstruktion spricht Lingnau [94] auch vom Variantenprinzip. Durch die Anwendung solcher Ansätze kann viel Geld und Zeit bei der Entwicklung neuer Produkte eingespart werden [2, 121]. Jedoch kann nach Oster [110] die Verwendung der genannten Ansätze zu einer enormen Steigerung der Variantenvielfalt führen. Beispielsweise lieferte das Mercedes-Benz-Werk Rastatt bereits zwischen 2004 und 2005 1,1 Millionen A-Klassen aus, von denen nur zwei identisch waren [137]. Damit gelangt die Überschaubarkeit der Produkte langsam an ihre Grenzen. Sicherlich ist das ein extremes Beispiel. Trotzdem führt es vor Augen, wie essentiell wichtig die gezielte Beherrschung der Variantenvielfalt ist. Auch wenn das Problem der Variantenvielfalt heute oft im Kontext der Produktentwicklung diskutiert wird, besteht es analog in anderen ingenieurtechnischen Domänen ebenfalls. Als Beispiel führt das Bundesministerium für Bildung und Forschung [103] unter anderem die Anwendungsbereiche des Maschinen- und Anlagenbaus sowie der Automatisierungs- und Fertigungstechnik auf. Letztere stehen gegenwärtig durch die sogenannte Energiewende und die damit einhergehende Variabilität bei der energetischen Auslegung fertigungstechnischer Prozessketten beim Produktionsplanungsprozess vor ganz neuen Herausforderungen [87, 112, 140,141, 144].

Zur Beherrschung der Variantenvielfalt fordert Holdschick bereits ein Variantenmanagement in der modellbasierten Produktlinienentwicklung [67] und damit den Einsatz der M&S. Unter einer Produktlinie versteht Withey [179] unterschiedliche Produkte mit einer gemeinsamen Menge an Funktionalitäten, die ein spezifisches Bedürfnis eines ausgewählten Marktes zufriedenstellen. Im Bereich der Softwaretechnik wird der Begriff Produktlinie zu Software-Produktlinien [152] erweitert und im Bereich der M&S spricht man von Modellfamilien [184, 25]. Übergeordnet sprechen Pohl et al. [121] auch von einer Systemfamilie. Allgemein wird nachfolgend eine Systemfamilie synonym mit Produkt- sowie Softwareproduktlinie verwendet.

Als Beispiel für eine Produktlinie kann eine Menge unterschiedlicher Fahrzeuge angeführt werden, deren Grundfunktionalität gleich ist, sich jedoch in Eigenschaften wie Farbe, Motorleistung, Reifen, Steuergerät, etc. unterscheiden. Für den Einsatz der M&S im Entwicklungsprozess bedeutet das, dass die M&S im Wesentlichen auch auf alle Systeme der zugehörigen Systemfamilie anzuwenden ist.

Bevor auf die M&S im Kontext von Systemfamilien näher eingegangen wird, soll zunächst der Begriff der Variabilität eingeführt werden.

2.3.1 Systemfamilien und Variabilität

Jézéquel [72] definiert Variabilität als die Annahme über die Unterschiede einzelner Individuen einer Familie. Übertragen auf diese Arbeit bedeutet Variabilität die Annahme über die Unterschiede einzelner Systeme einer Systemfamilie. In der Softwaretechnik gliedern Pohl et al. [121] die Variabilität in vier unterschiedliche Kategorien. Hierbei sprechen die Autoren ganz abstrakt von Artefakten, welche im Kontext dieser Arbeit als System bezeichnet werden.

- **Örtliche Variabilität:** Die unterschiedliche Ausprägung eines Systems zu einem Zeitpunkt.
- **Zeitliche Variabilität:** Die unterschiedliche Ausprägung eines Systems über der Zeit.
- **Externe Variabilität:** Für den Kunden/Auftraggeber sichtbare Variabilität eines Systems, die durch den Kunden/Auftraggeber beeinflusst werden kann.
- **Interne Variabilität:** Für den Kunden/Auftraggeber nicht sichtbare Variabilität des Systems, die auch nicht durch den Kunden/Auftraggeber beeinflusst werden kann.

Diese Arbeit greift nur die örtliche und interne Variabilität auf. Eine konkrete Ausprägung der örtlichen Variabilität wird Variante genannt. Wenn nicht weiter gekennzeichnet, wird nachfolgend in diesem Kapitel der Begriff Variante mit Simulationsmodellvariante gleichgesetzt.

Für eine Beherrschung der Variantenvielfalt setzen Schäuffele und Zurawka [136] ein notwendiges holistisches Variantenverständnis voraus. Darunter verstehen sie ein Verständnis über die Gemeinsamkeiten, Schnittmengen und Unterschiede aller Systeme einer Systemfamilie. Nach Strauch [172] kann dies durch eine *Variantenanalyse* erreicht werden. Weiterhin ist nach Jézéquel [72] eine Beschränkung der Variabilität über die Iden-

tifikation von Zwangsbedingungen innerhalb einer Systemfamilie notwendig, um Fehlvarianten zu vermeiden. Abbildung 7 zeigt schematisch den Prozess der Variantenanalyse in Anlehnung an Schäuffele und Zurawka [136].

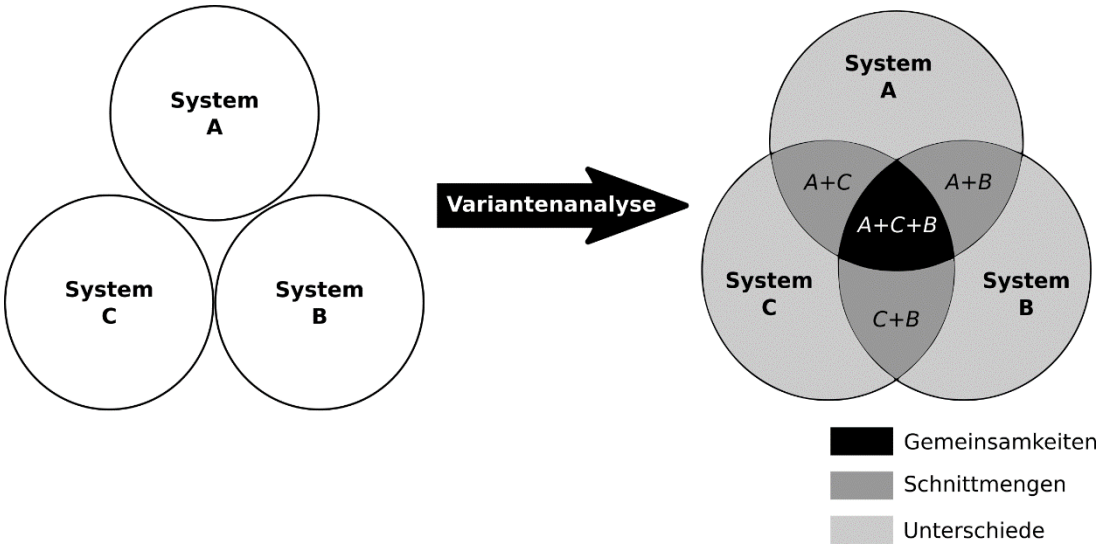


Abbildung 7: Prozess der Variantenanalyse in Anlehnung an Schäuffele und Zurawka [136].

Die Variabilität in einer Systemfamilie wird nach Halmans und Pohl [59] allgemein durch Variationspunkte beschrieben. Diese kennzeichnen unterschiedliche Systemvarianten. Holdschick [68] hebt hervor, dass die Variantenanalyse nur auf existierende Variantenstrukturen anzuwenden ist. Wie im Unterabschnitt 2.1.2 diskutiert, erfolgt bei der M&S eine strenge Trennung zwischen der Modelldynamik und der Modellstruktur. Demnach sind beim Variantenmanagement in der M&S in erster Linie die Varianten der Modellstrukturen zu betrachten, das heißt:

- die Modellkomponenten mit ihren wohldefinierten Schnittstellen,
- die horizontale und hierarchische Anordnung der Modellkomponenten sowie
- die Kopplungsbeziehungen zwischen den Modellkomponenten (EIC, IC, EOC).

Bei den Kopplungsbeziehungen wird in der M&S nach Zeigler et al. [187] zwischen externen Eingangskopplungen (EIC), externen Ausgangskopplungen (EOC) und internen Kopplungen (IC) unterschieden. Im Sinne eines effizienten Variantenmanagements ist Variabilität bezogen auf die Modelldynamik durch entsprechend konfigurierbare Modellkomponenten oder durch die Modellierung unterschiedlicher Modellkomponenten abzubilden. Im Kontext des in Abbildung 1, im Abschnitt 2.1, eingeführten Vorgehensmodells ist die Variantenanalyse als integraler Bestandteil der Systemanalyse einer Systemfamilie zu betrachten. Demzufolge muss gemäß Abbildung 1 das Konzeptmodell auch die Variabilität einer Systemfamilie spezifizieren.

Im Anschluss an die Systemanalyse kann wie in Holdschick [68] gefordert eine Formalisierung der Varianten, nachfolgend als *Variantenformalisierung* bezeichnet, erfolgen. Wie bereits im Unterabschnitt 2.1.2 erwähnt, sehen beispielsweise Wenzel et al. [178] bereits die Formalisierung einer Variante bezüglich des zeitlichen Aufwandes kritisch. Somit ist auch die Variantenformalisierung einer Menge unterschiedlicher Varianten, bezüglich des zeitlichen Aufwandes zu hinterfragen. Im Gegensatz dazu spricht sich

Holdschick [68] jedoch strikt für eine Variantenformalisierung unter Verwendung der Aussagenlogik aus, weil damit eine maschinelle Analyse der Modelleigenschaften sowie eine maschinelle Prüfung der Konsistenzbedingungen möglich werden.

Aus Sicht des Autors dieser Arbeit sollte eine Variantenformalisierung den Folgeschritt der Implementierung (vgl. Abbildung 1), im Weiteren *Variantenimplementierung* genannt, weitgehend unterstützen.

In der Literatur wird die Variantenimplementierung auch unscharf als Modellierung bezeichnet, Schmidt et al. [143]. Hierbei werden die einzelnen Varianten in der entsprechenden M&S-Umgebung softwaretechnisch codiert. Dieser Arbeitsschritt variiert in der Literatur je nach Ansatz. Somit wird eine detaillierte Diskussion erst in den nachfolgenden Unterabschnitten 2.3.2 und 2.3.3 durchgeführt.

Im Anschluss an die Implementierung erfolgt die *Variantengenerierung*, Alt [2]. Dabei wird eine gültige und ausführbare Variante durch die Auflösung aller vorhandenen Variationspunkte abgeleitet. Nach der Variantenimplementierung liegen die einzelnen Komponenten und Variantenstrukturen in der Regel in einer computerverarbeitbaren Form vor. Dadurch können nach Neyrick et al. [107] die konkreten Varianten auch automatisiert oder teilautomatisiert generiert werden. Liegt im Sinne der M&S eine gültige Variante in Form eines Simulationsmodells vor, so können jetzt Simulationsexperimente durchgeführt werden.

Im Kontext von M&S Projekten wurden in der Literatur verschiedene Ansätze zur Beherrschung der Variantenvielfalt identifiziert, wobei sich folgende drei Methoden: (i) *150% Modellierung*, (ii) *Featuremodellorientierter Ansatz* und (iii) ein *wissensbasierter Ansatz zur automatischen Adaption von Simulationsmodellen* hervorhoben, welche nachfolgend näher betrachtet werden.

2.3.2 Die 150% Modellierung

Die Grundidee der 150% Modellierung basiert auf der Varianteninplementierung in Form eines 150% Modells, nachfolgend als 150% Simulationsmodell bezeichnet. Das heißt, die identifizierte Variabilität der in Betracht zu ziehenden Varianten einer Systemfamilie werden in einem einzigen modular-hierarchischen 150% Simulationsmodell implementiert [136, 57]. Somit beschreibt, verwaltet und implementiert ein 150% Simulationsmodell grundlegend mehr Informationen als beispielsweise für eine bestimmte simulationsbasierte Untersuchung notwendig ist.

Abbildung 8 zeig ein Beispiel einer Variantenimplementierung in einem 150% Simulationsmodell. *Variante 1* und *Variante 2* stellen unterschiedliche Prozessketten zur Herstellung einer Werkstückart dar. Die Werkstücke werden im Generator (*G*) erzeugt, durchlaufen einige Puffer und Bearbeitungsstationen (Svr_i) und werden nach der Bearbeitung in der Senke (*T*) terminiert.

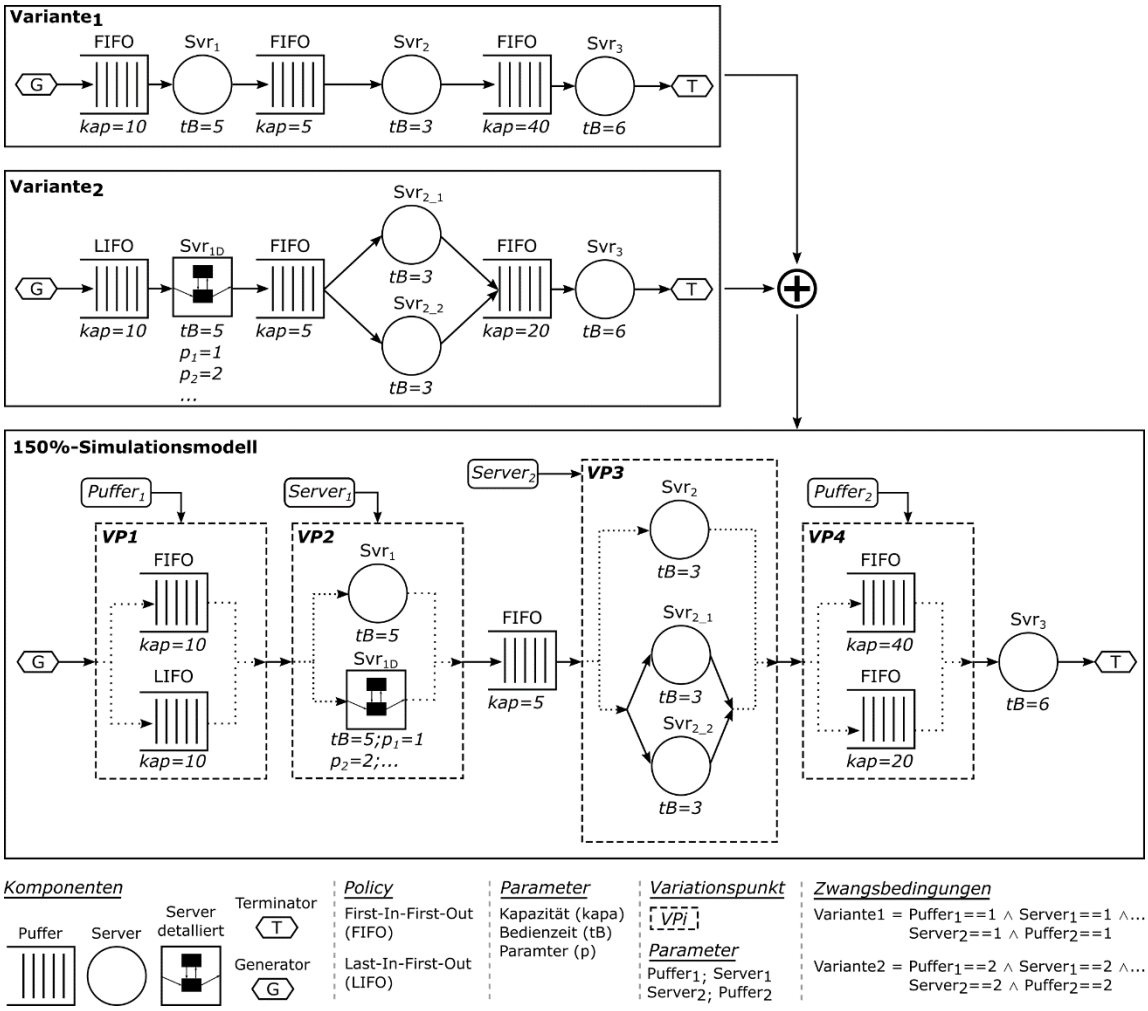


Abbildung 8: Beispiel unterschiedlicher Varianten einer Prozesskette und deren Abbildung in einem 150% Simulationsmodell.

Die Beiden Prozesskettenvarianten weisen folgende Variabilität auf:

1. Modelldynamik: Policy der Puffer (*FIFO* vs. *LIFO*)
2. Modelldynamik: Detaillierungsgrad von *Svr1* und *Svr1D*
3. Anzahl der Komponenten: Variante 2 besitzt zwei Bedienstationen vom Typ *Svr2* (*Svr2_1* und *Svr2_2*)
4. Modeparameter: Server *Svr1* bzw. *Svr1D* und Puffer vor *Svr3*
5. Kopplung der Komponenten: *Svr2* bzw. *Svr2_1* und *Svr2_2*

Das 150% Simulationsmodell in Abbildung 8 vereinigt beide Varianten. Die Variationspunkte (VP1-VP4) kennzeichnen die Variabilität und die Auswahlmöglichkeiten. Jedem Variationspunkt wird ein Parameter als Einflussgröße zugeordnet [53, 176, 34, 83, 56]. Im Beispiel heißen diese *Puffer1*, *Server1*, *Server2* und *Puffer2*. Mit den Variablen erhält man parametrierbare Variationspunkte, welche die örtliche Variabilität auf Parameter abbilden. Diese Parameter sind zum einen für die Beschreibung von Restriktionen der Variabilität und zum anderen für die Variantengenerierung notwendig. Restriktionen können mit den Parametern in Form von Zwangsbedingungen definiert werden, die die Auswahlmöglichkeiten eingrenzen. Im Beispiel in Abbildung 8 stellt jeder Variationspunkt zwei Auswahlmöglichkeiten zur Verfügung. Demnach können theoretisch aus dem 150% Simulationsmodell 2^4

unterschiedliche Varianten erzeugt werden, was nicht die Ausgangsproblemstellung widerspiegelt. Durch die Definition der Zwangsbedingungen, in Abbildung 8 rechts unten, wird die Anzahl der zu generierenden Varianten auf die mögliche Auswahl von genau zwei Varianten eingegrenzt, was der Ausgangsproblemstellung entspricht.

Die Variantengenerierung erfolgt durch Auflösung aller Variationspunkte auf Basis der aktuellen Wertzuweisungen an den zugehörigen Parametern. Dadurch werden die nicht gewählten Komponenten des Simulationsmodells deaktiviert. Nach der Generierung einer gültigen Variante können Simulationsexperimente, im Sinne einzelner Simulationsläufe, durchgeführt werden. Jedoch verbleiben auch die deaktivierten Komponenten als *toter Code* im Simulationsmodell.

Haber et al. [57] kritisieren das Variantenmanagement mittels 150% Simulationsmodellen scharf. Diese ist, ergänzt um eigene Anmerkungen des Autors, nachfolgend in sechs Punkten zusammengefasst.

1. Die Modellierung der unterschiedlichen Varianten innerhalb eines 150% Simulationsmodells führt zu unübersichtlichen und komplexen modular-hierarchischen Simulationsmodellen.
2. Bei steigender Komplexität des 150% Simulationsmodells wird das Debugging und nach Ansicht des Autors allgemein die Wartung kompliziert.
3. Die Variationspunkte können auf verschiedenen Ebenen eines modular-hierarchischen Simulationsmodelles auftreten, was die Übersicht bezüglich der implementierten Varianten erheblich erschwert.
4. Modifikationen der Variationspunkte oder Komponenten auf unterschiedlichen hierarchischen Ebenen können zu unerwarteten Problemen führen.
5. Die Spezifikation der Variabilität und Funktionalität innerhalb eines dynamischen Simulationsmodells verstößt gegen das Konzept der *Trennung der Zuständigkeiten* in der Informatik.
6. Die Wiederverwendbarkeit von 150% Simulationsmodellen stößt mit zunehmender Komplexität schnell an Grenzen.

Trotz dieser eindeutigen Kritikpunkte wird die 150% Modellierung insbesondere im Kontext von Softwareproduktlinien [177] und bei Applikationen in der MATLAB/Simulink-Umgebung eingesetzt, wie zum Beispiel in [176, 34, 83, 56] gezeigt. Die praktische Bedeutung der 150% Modelle ist vermutlich mit dem relativ einfachen Übergang von der Variantenmodellierung über die Variantenimplementierung bis zur Variantengenerierung in einer M&S-Umgebung begründbar.

Im nachfolgenden Abschnitt wird ein Variantenmanagementansatz aus der Softwaretechnik präsentiert, welcher die 150% Modellierung mit einem Featuremodell kombiniert. Mit dem Featuremodell wird eine visuelle Beschreibung der Variationspunkte unter Verwendung eines Graphen ermöglicht.

2.3.3 Featuremodellorientierter Ansatz

Zur Lösung des Problems des Variantenmanagements in der Softwaretechnik plädiert Alt [2] für einen featuremodellbasierten Variantenbildungsansatz. Dieser ist in Abbildung 9 gezeigt und kann auf das Gebiet der M&S übertragen werden. Cavarlé et al. [17] beschreiben einen

solchen Ansatz als eine Kombination von Feature-Oriented Development (FOD) und Model-Driven Development (MDD), woraus in der Softwaretechnik der Term Feature-Oriented Model-Driven Development (FOMDD) abgeleitet wurde.

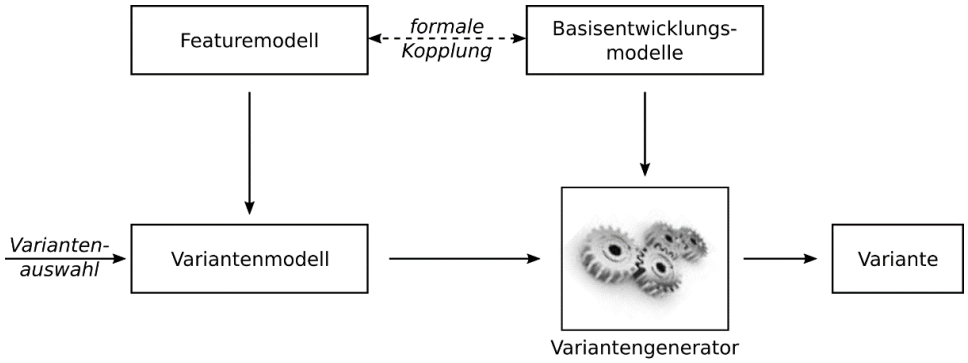


Abbildung 9: Prinzip der featuremodellbasierten Variantenbildung modifiziert aus [2].

MDD basiert auf der Idee, Modelle unter Verwendung von Transformationsmethoden (teil)automatisch in andere Modelle oder in ausführbaren Code zu überführen und bildet heute nach Hussmann et al. [69] einen wesentlichen Bestandteil der systematischen und effizienten Softwareentwicklung. Wie die Recherchen von Cetinkaya et al. [19] und Durak et al. [31] zeigen, zieht die MDD Methodik sukzessive in die M&S ein. FOD ist ein Paradigma der Softwaretechnik und beschäftigt sich mit der Erstellung, Anpassung sowie der Synthese von umfangreichen Softwaresystemen unter Verwendung von *Feature-modellen* [3]. Featuremodelle (FM), auf Deutsch Merkmalmodelle genannt, spezifizieren alle Merkmale einer Systemfamilie und deren Abhängigkeiten untereinander für einen bestimmten Anwendungsbereich. Sie wurden ursprünglich von Kang et al. [74] im Rahmen der featureorientierten Domänenanalyse eingeführt. Die Domänenanalyse kann mit dem Prozess der Variantenanalyse gleichgesetzt und demnach ein FM mit einem Konzeptmodell verglichen werden.

Ein FM ermöglicht eine hierarchische Strukturierung der Merkmale in Form eines gerichteten azyklischen Graphen. Die Knoten repräsentieren die spezifischen Merkmale und die Kanten die Zusammenhänge zwischen den jeweiligen Merkmalen, Ryssel [130]. Ein FM ermöglicht die Spezifikation der Variabilität einer Systemfamilie auf einem höheren Abstraktionsniveau. Somit können komplexe Zusammenhänge übersichtlich und verständlich spezifiziert werden.

Der Ansatz nach Alt [2], gemäß Abbildung 9, beinhaltet die *Basisentwicklungs-modelle* aller möglichen Systemkomponenten der Systemfamilie. Im Kontext der M&S entspricht die Komponente der Basisentwicklungsmodelle einem 150% Simulationsmodell, welches alle notwenigen dynamischen Komponenten und deren Kopplungen untereinander abbildet. Durch die *formale Kopplung* zwischen einem FM und den Basisentwicklungs-modellen werden die Features und Variationspunkte des FM den Komponenten und Variationspunkten des 150% Simulationsmodells eindeutig zugeordnet. Abbildung 10 zeigt plakativ ein FM bezüglich der in Abbildung 8 eingeführten Problemstellung. Weiterhin ist zur Illustration aller Beschreibungsmittel eines FMs ein zusätzliches Merkmal *Scope* aufgeführt, welches nicht im Beispiel in Abbildung 8 enthalten ist.

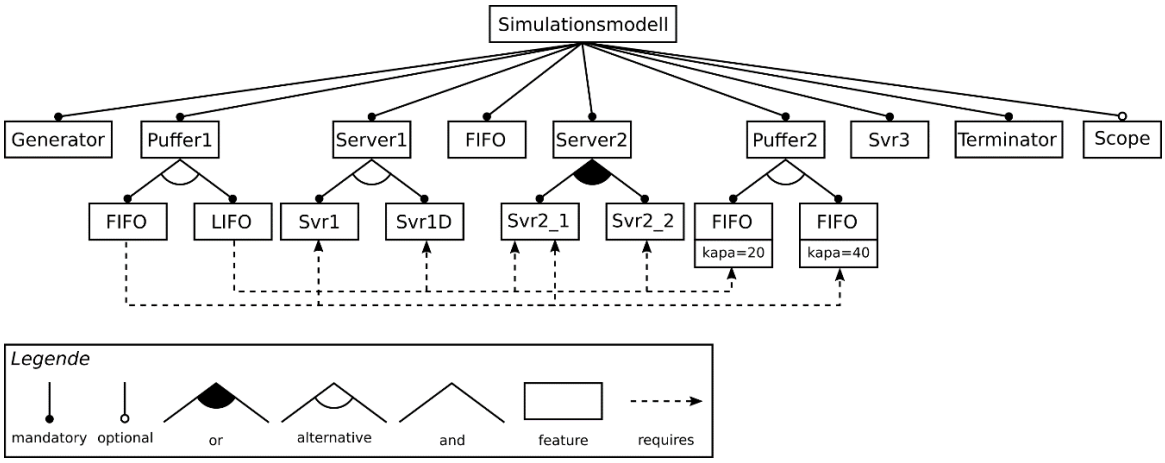


Abbildung 10: FM zur Problemstellung in Abbildung 8.

Die Zusammenhänge der einzelnen Merkmale untereinander, dargestellt mit Kanten, sind gemäß Batory [10] und Karatas et al. [75] wie folgt definiert:

- *Mandatory*: Feature ist zwingend notwendig
- *Optional*: Feature ist optional
- *Alternative*: Nur ein Subfeature ist auszuwählen
- *Or*: Ein oder mehrere Subfeatures sind auszuwählen
- *And*: Alle Subfeatures sind zu wählen
- *Requires*: Die Auswahl von Feature A impliziert die Auswahl von Feature B

Wie in Abbildung 10 zu erkennen ist, bilden die Modellkomponenten und deren Parameter (vgl. Abbildung 8) die wesentlichen Merkmale. Aus Gründen der besseren Übersicht sind im FM in Abbildung 10 nur an zwei Merkmalen die Parameter angegeben. In Abbildung 10 kennzeichnen die Kanten *Or*, *Optional* und *Alternative* die Variationspunkte zur Auswahl einer konkreten Variante. Zur Einschränkung der Auswahlmöglichkeiten können analog dem Beispiel in Abbildung 8 Zwangsbedingungen mittels *Requires* definiert werden. Dieses Beschreibungsmittel reduziert die Anzahl der möglichen Varianten in Abbildung 10 auf genau vier. Die zwei zusätzlichen Varianten folgen aus der optionalen Auswahl des Merkmals *Scope*. Im Beispiel kann dieses Merkmal zur Visualisierung von Simulationsergebnissen ausgewählt werden. Die zwei Variationspunkte an den Merkmalen *Puffer1* und *Server1* werden durch die *Requires* Zwangsbedingung eingeschränkt. Werden beispielsweise die Variationspunkte *FIFO* oder *LIFO* im *Puffer1* aufgelöst, so erfolgt die Auswahl der zugehörigen Features an den Knoten *Server1*, *Server2* und *Puffer2* gemäß den Zwangsbedingungen.

Durch die hierarchische Anordnung der Features im FM steigt die Übersicht gegenüber einem 150% Simulationsmodell deutlich an.

Das FM entspricht grundlegend einem Konzeptmodell, welches alle Varianten darstellt. Die Implementierung des FM erfolgt gemäß Abbildung 9 in einem *Variantenmodell*. Nach Alt [2] besteht der Unterschied zwischen einem FM und einem Variantenmodell darin, dass letzteres Schalter zum gezielten Ein- und Abschalten von Features an Variationspunkten definiert, die eine konkrete Variantenauswahl ermöglichen. Somit erfolgt eine konkrete Variantenauswahl ausschließlich im Variantenmodell.

Die Variantengenerierung wird durch den *Variantengenerator* realisiert. Dieser erhält als Eingabedaten ein Variantenmodell mit eindeutig aufgelösten Variationspunkten, sowie das zugehörige 150% Simulationsmodell. Letzteres entspricht der Komponente Basisentwicklungsmodelle in Abbildung 9 wie zuvor beschrieben. Die formale Kopplung zwischen dem FM und dem 150% Simulationsmodell gilt auch für das Variantenmodell. Somit kann der Variantengenerator anhand der Informationen des Variantenmodells die Variationspunkte des 150% Simulationsmodells auflösen und damit eine konkrete Simulationsmodellvariante aktivieren. Analog der 150% Modellierung verbleiben die nicht gewählten Komponenten als deaktivierte Komponenten und damit als *toter Code* im Simulationsmodell.

Eine praktische Umsetzung des beschriebenen Ansatzes stellt das Softwaresystem *pure::variants* [78] dar. Für die M&S-Umgebung MATLAB/Simulink existiert die Spezialausprägung *prure:variants for Simulink* [122]. Steiner et al. [153] kritisieren jedoch den Ansatz von *pure:variants* for Simulink, weil der *tote Code* in der generierten Variante verbleibt. Den Verbleib von totem Code im generierten Simulationsmodell nennen Sie einen Entwurfsfehler, welcher zu beseitigen ist. Sie präsentieren den Ansatz *Hephaestus*, welcher ein *Refactoring* der erzeugten Variante ermöglicht und den *toten Code* ohne Änderung der Semantik entfernt. Auch TheMathWorks liefert für ihre MATLAB/Simulink Umgebung inzwischen Methoden für das Variantenmanagement [52, 165]. Hierbei kann ein 150% Simulationsmodell beispielsweise mittels *Variant Subsystems* modelliert werden. Ein *Variant Subsystem* kennzeichnet die Variabilität und ermöglicht für die Variantenauswahl die Zuordnung von speziellen Parametern gemäß Unterabschnitt 2.3.1. Für eine übersichtliche Darstellung der Variabilität kann auf den sogenannten *Variant Manager* zurückgegriffen werden. Dieser fast die Struktur in Form eines Graphen, welcher an ein FM angelehnt ist, und die Parametrierungsmöglichkeiten des 150% Simulationsmodells übersichtlich zusammen. Weiterhin können Zwangsbedingungen als Restriktion für die Variantengenerierung spezifiziert werden.

Ein essentieller Vorteil der diskutierten Ansätze, die sich konzeptionell an das Prinzip in Abbildung 9 anlehnen, ist die Bewältigung der Variantenvielfalt mittels höherer Abstraktion im Gegensatz zur reinen 150% Modellierung. Durch die abstrakte und hierarchische Strukturierung aller Features und Variationspunkte in einem FM steigen die Übersichtlichkeit und das Variantenverständnis erheblich. Weiterhin ermöglicht der Variantengenerator eine Steigerung der Flexibilität und der Effizienz hinsichtlich einer automatischen Variantengenerierung. Der Nachteil dieser Ansätze bei der bisherigen Anwendung in der M&S ist die bleibende Verwendung eines 150% Simulationsmodells. Insofern bleiben die Kritikpunkte einer 150% Modellierung, welche im vorherigen Abschnitt diskutiert wurden, zum großen Teil bestehen. Die zuvor positiv hervorgehobene bessere Übersichtlichkeit des Variantenverständnisses durch das FM bedeutet aber auch, dass die Wartung und das Debugging des FM hinzukommen. Weiterhin muss stets die Konsistenz zwischen dem FM und dem 150% Simulationsmodell abgesichert werden. Jeder Änderung des FM folgt eine entsprechende Änderung im 150% Simulationsmodell und umgekehrt. Jegliche Inkonsistenz führt zu einer fehlerhaften Variantengenerierung. Somit stellt der in Abbildung 9 gezeigte Ansatz für die M&S bezüglich der bisher diskutierten Realisierungen keine optimale Lösung des Variantenmanagements dar. Die 1:1 Übertragung dieses Ansatzes auf die M&S ist auf jeden Fall zu hinterfragen. Darüber hinaus fokussieren sich die bisherigen Arbeiten nur auf

die Variantenbeherrschung bei der Modellbildung und die Simulationsmodellgenerierung. Die gemäß Abbildung 1 anschließende Phase der Simulation und die damit verbundene Durchführung von Simulationsexperimenten werden nicht adressiert. Im nächsten Abschnitt wird der *wissensbasierte Ansatz zur automatischen Adaption von Simulationsmodellen* nach Lattner et al. [88] diskutiert.

2.3.4 Wissensbasierter Ansatz zur automatischen Adaption von Simulationsmodellen

Im Anwendungsfeld der Produktion und Logistik präsentieren Lattner et al. [88] einen wissensbasierten Ansatz zur automatischen Adaption von Simulationsmodellen. Dieser ist in Abbildung 11 gezeigt. Er ermöglicht neben der klassischen Variation der Einflussgrößen auch eine automatische Variation der Struktur eines Simulationsmodells. Die Kernidee beruht auf der automatischen Adaption eines unvollständigen Simulationsmodells, das Lattner et al. [88] *basic model* nennen. Dieses implementiert nur die Gemeinsamkeiten aller Varianten möglicher Simulationsmodelle und bildet damit den Grundrahmen für die Variantengenerierung. In Analogie zur 150% Modellierung wird vom Autor der vorliegenden Arbeit der Term des 60%-Modells eingeführt. Auf Basis des 60%-Simulationsmodells werden unter Verwendung einer Wissensbasis automatisiert ein oder mehrere 100%-Simulationsmodelle generiert. Damit stellt der Ansatz eine alternative Form des Variantenmanagements dar.

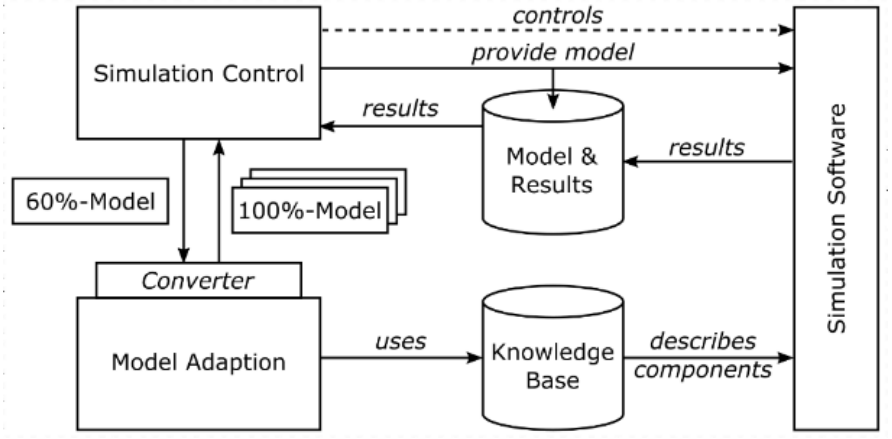


Abbildung 11: Ansatz zur automatischen wissensbasierten Adaption von Simulationsmodellen modifiziert aus [88].

Gemäß Abbildung 11 basiert der Ansatz auf fünf wesentlichen Softwarekomponenten: (i) *Simulation Software*, (ii) *Simulator Control*, (iii) *Model Adaption*, (iv) *Knowledge Base* und (v) *Model & Results*. Die Komponente *Simulation Software* entspricht einer ereignisdiskreten M&S-Umgebung. Die *Simulation Control* verwaltet das *60%-Modell* sowie alle bereits generierten *100%- Modelle*. Weiterhin steuert die *Simulation Control* die Adaption der Modelle durch den *Model Adapter*, initialisiert die *Simulation Software* und sie startet und bewertet die Simulationsexperimente. Die *Knowledge Base* beinhaltet das relevante Wissen zur *Simulation Software* und zur Adaption des 60%-Modells zu 100%- Modellen. Das heißt, sie beschreibt die Simulationsmodellkomponenten und deren Abhängigkeiten unter einander zur Adaption des 60%-Modells, die zulässigen Parametrierungen der Komponenten sowie die möglichen Kopplungsbeziehungen. Damit speichert die *Knowledge Base*

das Wissen der Variantenanalyse in Form von konkreten Variantenspezifikationen. Das Wissen wird unter Verwendung der logischen Programmiersprache *Prolog* [23] in Form von *Fakten* und *Regeln* deklarativ beschrieben. Die Komponente *Model Adaption* generiert unter Verwendung der *Knowledge Base* eine oder mehrere gültige Varianten von 100%- Simulationsmodellen. Die Adaption des 60%-Simulationsmodells basiert auf drei wesentlichen Operationen: (i) *Hinzufügen von Komponenten,* (ii) *Ersetzen von Komponenten durch kompatible Komponenten* und (iii) *Parametrierung von Komponenten.* Die Adaption der Kopplungsbeziehungen erfolgt automatisch auf Basis der ausgewählten Komponenten des Simulationsmodells und dem dazu gespeicherten Wissen. Die Softwarekomponente *Model & Results* speichert die generierten 100% Modelle sowie die zugehörigen Simulationsergebnisse. Die notwendigen Transformationen in ein durch die *Model Adaption* ausführbares Format und eine Rücktransformation in das *Simulation Software* Format erfolgen durch den *Converter*. Die Ausführung einer Simulationsmodellvariante und die Bewertung der Simulationsergebnisse erfolgt wie eingehend erläutert durch die *Simulation Control*. Demgemäß entscheidet diese auch, ob nach Abschluss eines Simulationslaufes weitere Simulationsläufe mit einer Simulationsmodellvariante auszuführen sind.

Zusammenfassend sei vermerkt, dass die Variantengenerierung in Form von 100%-Modellen auf Basis eines gemeinsamen 60%-Modells und einer konkreten Variantenspezifikation erfolgt. Einen ähnlichen Ansatz beschreiben Wagner et al. [173] unter dem Begriff *Ontology Driven Simulation*. Für die Spezifikation des Wissens greifen die Autoren auf eine Ontologie zurück. Eine Ontologie ist nach Gruber [54] eine explizite Spezifikation eines Gegenstandsbereiches. Die Methode ist insbesondere vom *Semantic Web* bekannt.

Ein Vorteil des Ansatzes in Abbildung 11 gegenüber den vorherigen Ansätzen ist, dass die Nachteile der 150% Modellierung, die im Abschnitt 2.3.1 diskutiert wurden, entfallen. Allerdings sind bei diesem Ansatz Kenntnisse bezüglich der logischen Programmierung in Prolog zum Aufbau einer *Knowledge Base* notwendig. Die logische Programmierung erfolgt textuell und setzt ein Verständnis der Prädikatenlogik voraus. Die Verständlichkeit komplexer Zusammenhänge gegenüber einer graphischen Darstellung sinkt. Auch besitzen Ingenieure oft keinen hinreichenden Zugang zur logischen Programmierung.

2.4 Zusammenfassung

Ausgehend von den Grundlagen der M&S wurden fünf allgemeine Kategorien von Experimentzielen identifiziert: (i) *Validierung, Verifikation & Testing*, (ii) *Explorative Analyse*, (iii) *Screening*, (iv) *Sensitivitätsanalyse* und (v) *Optimierung*. Für jedes Experimentziel, außer der explorativen Analyse, existieren eine Vielzahl numerischer Methoden mit umfassenden Parametrierungsmöglichkeiten. Die numerischen Methoden werden im Rahmen der Arbeit auch Experimentmethoden genannt. Diese Vielfalt bedeutet für den Anwender beim Durchführen von M&S Projekten eine enorme Herausforderung. Neben Literaturrecherchen bleibt nach Ansicht des Autors nur die Möglichkeit des *systematischen Ausprobierens*. Das heißt, bezüglich einer komplexen Problemstellung müssen eine Menge unterschiedlicher Simulationsexperimente manuell aufgebaut und möglichst automatisiert ausgeführt werden. Für den Aufbau von Simulationsexperimenten wurden drei unterschiedliche Kategorien: (i) *einfach*, (ii) *komplex* und (iii) *hochkomplex* identifiziert. Dabei zeigte sich, dass der Aufbau

komplexer Simulationsexperimente bisher in der Literatur nur unzureichend und unterschiedlich spezifiziert ist.

Das systematische Ausprobieren hat das bekannte Problem der Variantenvielfalt zur Folge. Hier können Ansätze des Variantenmanagements Abhilfe schaffen. Das Variantenmanagement umfasst grundlegend eine: (i) *Variantenanalyse*, (ii) *Variantenformalisierung*, (iii) *Varianteninplementierung* und (iv) *Variantengenerierung*. Bei der M&S kommt das Variantenmanagement derzeit nur bei der Verwaltung von unterschiedlichen Simulationsmodellvarianten zum Einsatz und bezieht sich damit ausschließlich auf die Phase der Modellbildung. In diesem Zusammenhang wurden zuvor drei anerkannte Ansätze: (i) die *150% Modellierung*, (ii) der *featuremodellorientierte Ansatz* und (iii) ein *wissensbasierter Ansatz zur automatischen Adaption von Simulationsmodellen* diskutiert. Wie gezeigt, besitzen die diskutierten Ansätze einige grundlegende Schwächen oder Nachteile bezüglich des Debuggings, der Wartung, der Wiederverwendbarkeit, der Vermischung von Zuständigkeiten (Vermengung von Struktur und Dynamik), der Notwendigkeit von Kenntnissen prädikatenlogischer Programmierung und so weiter. Dadurch sind sie aus Sicht des Autors nur bedingt für ein praktikables Variantenmanagement bei M&S Projekten geeignet. Weiterhin berücksichtigen die drei Ansätze nur das Variantenmanagement für die Phase der Modellbildung. Die Phase der Simulation wird höchstens in Form eines Simulationslaufes thematisiert.

Das Ziel dieser Arbeit ist es eine Methode des Variantenmanagements zu entwickeln, die umfassend beide Phasen der Modellbildung und Simulation unterstützt. Eine mögliche Grundlage kann das in der Modellbildung bekannte, jedoch bisher nicht im Kontext des Variantenmanagements eingesetzte, System Entity Structure/Model Base Framework (SES/MB) bieten. Dieser Ansatz wird im nächsten Kapitel eingehend betrachtet.

3 Das System Entity Structure/Model Base Framework als Basis eines Variantenmanagements

In diesem Kapitel erfolgt eine Untersuchung, wie das von Zeigler eingeführte System Entity Structure/Model Base (SES/MB) Framework zum Variantenmanagement in der M&S verwendet werden kann. Wie in den vorhergehenden Kapiteln bereits eingeführt, soll dabei unter Variantenmanagement die Variantenmodellierung, Variantenimplementierung und Variantengenerierung verstanden werden. Zunächst erfolgt ein allgemeiner Überblick zum SES/MB Framework. Anschließend werden die Grundlagen des Frameworks an einem Beispiel aus dem Anwendungsbereich Produktion und Logistik präsentiert und bestehende Probleme aufgezeigt. Darauf aufbauend werden Lösungsvorschläge zur Überwindung aufgezeigter Probleme erarbeitet. Das Kapitel beschränkt sich auf das Variantenmanagement von Simulationsmodellen einer Systemfamilie und endet mit einer Zusammenfassung sowie Diskussion bezüglich einer Erweiterung des Variantenmanagements auf Simulationsexperimente.

3.1 Übersicht zum System Entity Structure/Model Base Framework

Die ersten Grundlagen des System Entity Structure/Model Base (SES/MB) Frameworks wurden von Zeigler [185] im Kontext der DEVS Theorie entwickelt. Das ursprüngliche Ziel war es, den Übergang vom Ansatz *modeling in the small* zum *modeling in the large* zu realisieren. Unter *modeling in the small* versteht man die Verwendung von Simulationsmodellen als Werkzeug zur Lösung genau eines Problems. Der *modeling in the large* Ansatz wird von Zeigler [185] als ein holistischer M&S Ansatz gesehen, bei dem vor allem die Wiederverwendung sowie Weitergabe von Simulationsmodellen an Dritte, eine einfache Modifikation und Kombination von Simulationsmodellen und die Kommunikation sowie Dokumentation von komplexen Simulationsmodellen zur Lösung unterschiedlicher Problemstellungen im Vordergrund stehen. Seit der Einführung durch Zeigler wird das SES/MB Framework sukzessiv durch verschiedene Autoren kontinuierlich weiterentwickelt [187, 116, 115, 188, 186, 133, 128, 141, 148, 114].

In der Literatur hat sich das SES/MB Framework bei der Lösung unterschiedlicher ingenieurtechnischer Problemstellungen bereit bewährt. So verwendete Hagendorf in [58] das SES/MB Framework im Kontext der Struktur- und Parameteroptimierung für das Modellmanagement gemäß Abbildung 6 im Abschnitt 2.2.2. Schwatinski et al. [149] verwendeten das SES/MB Framework für den Entwurf und die Realisierung flexibler aufgabenorientierter Robotersteuerungen. Schmidt et al. [139] verwenden das Framework im Kontext des *Model-Based Testings* in der MATLAB/Simulink Umgebung. Durak et al. [32] präsentieren einen Ansatz für die modellbasierte Entwicklung von Flugszenarien für den *Air VEhicle Simulator* (AVES) mittels des SES/MB Frameworks. Auf Basis des SES/MB Frameworks entwickelte Larek [87] eine Methode zur numerischen Minimierung des Ressourcenverbrauchs von fertigungstechnischen Prozessketten. Weiterhin führen Zeigler und Hammonds [186] die SES als eine Ontologie zur expliziten Spezifikation eines Gegenstandsbereiches im Kontext des *Data Engineerings* ein.
Mit dem SES/MB Framework ist ein Vorgehensmodell verbunden. Dieses illustriert schematisch Abbildung 12.

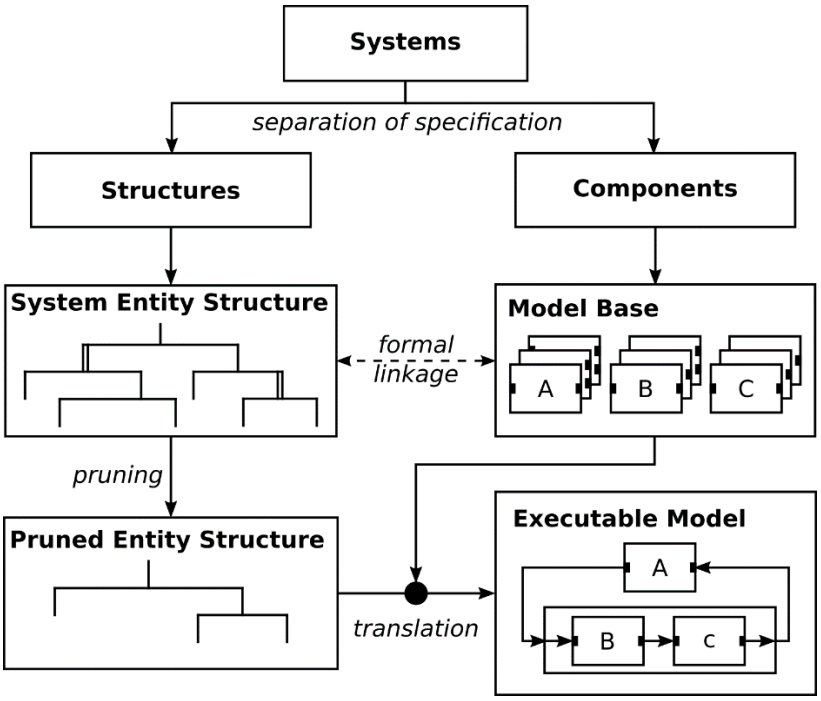


Abbildung 12: Vorgehensmodell zum SES/MB Framework modifiziert aus [187, 58].

Den Ausgangspunkt bildet eine Vielzahl unterschiedlicher Systeme, die in Anlehnung an Abschnitt 2.3 als Systemfamilie bezeichnet werden. Diese werden, wie bereits im zweiten Kapitel diskutiert, bezüglich der Systemstrukturen und der dynamikbeschreibenden Systemkomponenten analysiert und formalisiert. Somit findet keine Verwendung von 150%-Modellen statt. Die Systemkomponenten werden als wiederverwendbare und parametrierbare Komponenten mit eindeutigen Ein- und Ausgangsschnittstellen implementiert und in einer Modellbibliothek (*Model Base*) organisiert. Die formale Spezifikation der Systemstrukturen erfolgt in einer SES, welche durch einen gerichteten azyklischen Graphen dargestellt wird. Weiterhin muss die SES eine *formale Kopplung* zu den Komponenten der MB definieren. Ein Ansatz zur Umsetzung der Kopplung wird nachfolgend im Abschnitt zu den notwendigen Erweiterungen des Frameworks vorgestellt. Weiterhin wird in diesem Zusammenhang auch die Parametrierung referenzierter Systemkomponenten diskutiert. Zur automatisierten Ableitung eines konkreten ausführbaren Simulationsmodells definiert das SES/MB Framework die Methoden *puning* und *translation*. Pruning ermöglicht das Ableiten einer konkreten Struktur eines Simulationsmodells in Form einer *Pruned Entity Structure* (PES). Eine PES bildet nach Gleichung 9 eine Teilmenge einer SES.

$$PES \subseteq SES \tag{9}$$

Die *translation* Methode generiert auf Basis der PES und der Komponenten der MB ein ausführbares Simulationsmodell.

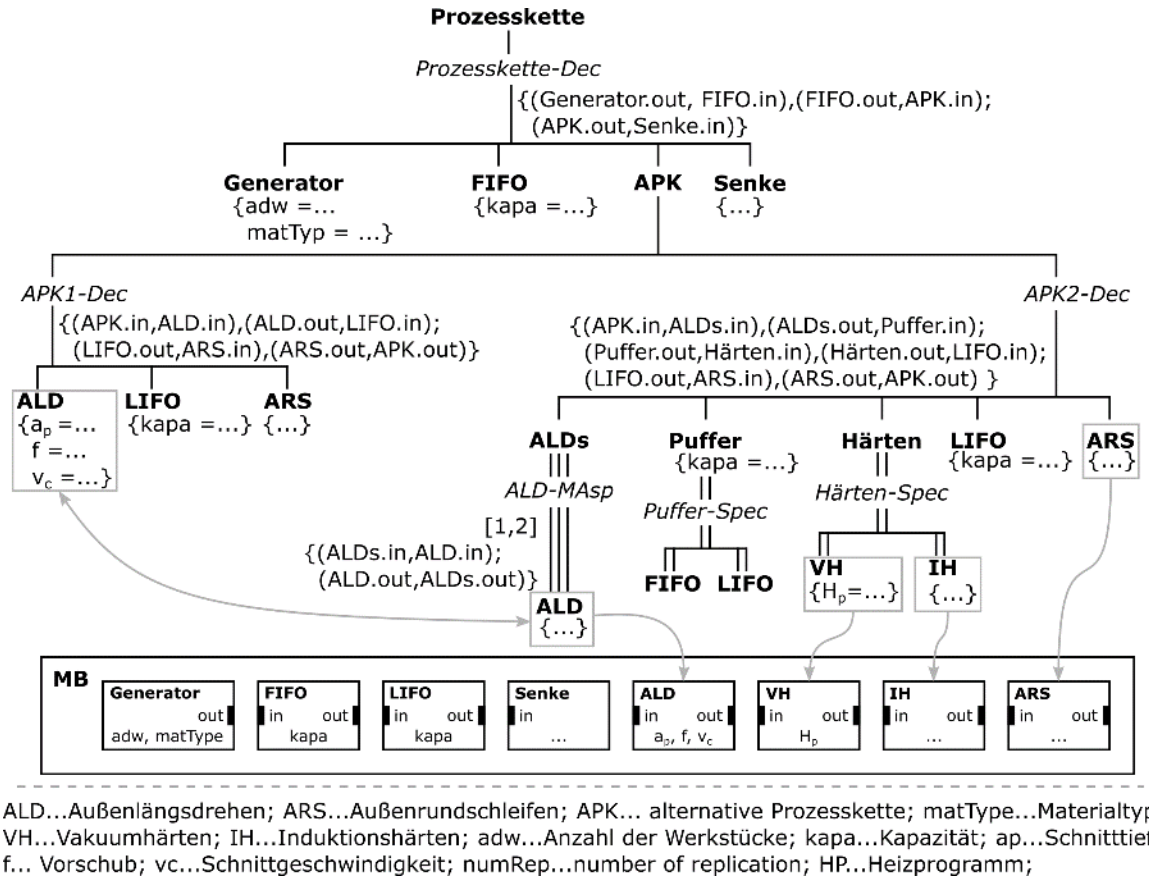
Allgemein können bereits jetzt erste Parallelen zum Variantenmanagement nach Abschnitt 2.3 gezogen werden. Demnach spezifiziert und implementiert die SES die Varianten der Systemstrukturen der zu betrachtenden Simulationsmodelle und, wie später gezeigt wird, auch die möglichen Parametrierungen der Systemkomponenten. Daneben unterstützt das SES/MB Framework mit den Methoden *pruning* und *translation* die Variantengenerierung bis hin zu ausführbaren Simulationsmodellen.

In Bezug auf das SES/MB Framework existieren in der Literatur zwei wesentliche Softwarelösungen: (i) die Tiny-SES-Toolbox für MATLAB/Simulink [148] und (ii) die MS4Me-Umgebung [188], die die Verwendung des SES/MB Frameworks in der M&S ermöglichen. Die erste Softwarelösung basiert maßgeblich auf der logischen Programmierung mit Prolog sowie einer implementierten Schnittstelle zwischen Prolog und MATLAB/Simulink. Die Spezifikation und Implementierung der SES erfolgt unter Verwendung vordefinierter Prädikate in Prolog. Dadurch ist die Softwarelösung, analog zu den Erläuterungen im Abschnitt 2.3.3, für ingenieurtechnische Problemstellungen unattraktiv. Die Softwarelösung MS4Me basiert auf der DEVS Theorie, gemäß Zeigler et al. [187], und unterstützt die automatische Generierung von DEVS-basierten Simulationsmodellen in der Java-Umgebung. Zur Spezifikation und Implementierung der SES und der Komponenten der MB wird von den Entwicklern eine textbasierte *natural language* mit einer abgegrenzten Menge an Anweisungen zur Verfügung gestellt. Jedoch leidet bei der textuellen Spezifikation der SES die Übersicht. Ebenso ist die DEVS Theorie im ingenieurtechnischen Anwendungsbereich fast unbekannt. Die Anwendung von MS4Me erfolgt vorwiegend im akademischen und militärischen Bereich in den USA.

Nach der allgemeinen Einführung werden im nächsten Abschnitt wesentliche Aspekte des SES/MB Frameworks detailliert diskutiert und an einem Beispiel aus dem Anwendungsbereich der Produktion und Logistik präsentiert. Zusätzlich werden erste Parallelen zum Variantenmanagement von Simulationsmodellen in Anlehnung an Abschnitt 2.3 hergestellt.

3.2 Grundlagen des SES/MB Frameworks

Abbildung 13 stellt die wesentlichen Gesichtspunkte einer SES mit zugehöriger MB an einem Anwendungsbeispiel dar. Die SES spezifiziert eine Menge unterschiedlicher Systemstrukturen einer fertigungstechnischen Prozesskettenfamilie in Form eines Baums. Die MB illustriert einige zugehörige Systemkomponenten mit ihren Ein- und Ausgabeschnittstellen sowie Konfigurationsparametern. Beispielsweise besitzt die dargestellte Komponente *ALD* eine Eingangsschnittstelle *in*, eine Ausgabeschnittstelle *out* sowie die konfigurierbaren Parameter *ap*, *f* und *vc*. Anhang A zeigt die prinzipielle Spezifikation der SES in Abbildung 13 unter Verwendung der *natural language* nach Zeigler und Sarjoughian [90]. Im Weiteren wird auf die Spezifikation und Interpretation der SES, also die Syntax und Semantik der einzelnen Knoten, Kanten und Attribute sowie die SES Axiome, eingegangen. Darüber hinaus wird die formale Kopplung zwischen der SES und den Komponenten in der MB diskutiert. Die Grundlagen zum SES/MB Framework in den nächsten Unterabschnitten basieren maßgeblich auf den Arbeiten von Zeigler et al. [187], Zeigler und Hammonds [186] sowie Zeigler und Sarjoughian [188]. Abschließend werden die Probleme aus Sicht des Autors analysiert.



ALD...Außenlängsdrehen; ARS...Außenrundschleifen; APK... alternative Prozesskette; matType...Materialtyp; VH...Vakuumhärten; IH...Induktionshärten; adw...Anzahl der Werkstücke; kapa...Kapazität; ap...Schnitttiefe; f... Vorschub; vc...Schnittgeschwindigkeit; numRep...number of replication; HP...Heizprogramm;

Abbildung 13: Beispielhafte Spezifikation unterschiedlicher Systemstrukturen einer Prozessketfamilie mit einer SES und Organisation der zugehörigen Systemkomponenten in einer MB.

3.2.1 Elemente und Axiome der SES

Die Taxonomie in Abbildung 14 klassifiziert die Knoten- und Kantentypen einer SES. Die Kantentypen stehen in unmittelbarem Zusammenhang mit den Knotentypen. Die Knotentypen *Abstraktion* und *Zeit* sind von Santucci et al. [133] im Kontext der Modellierung von unterschiedlichen Detaillierungsgraden (*level of detail*) mit der SES eingeführt worden. Diese sind vollständigkeitshalber in die Taxonomie aufgenommen, werden aber nicht weiter diskutiert.

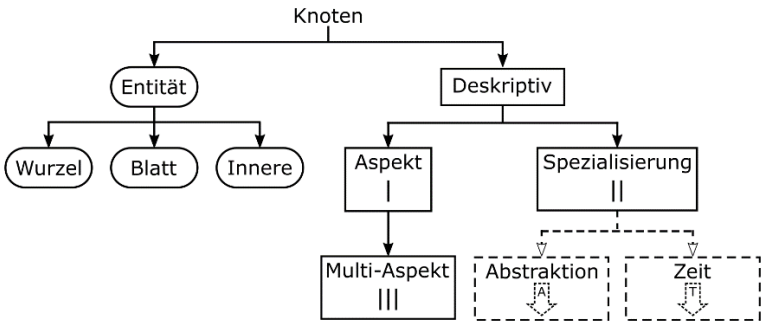


Abbildung 14: Taxonomie der Knoten- und zugehörige Kantentypen einer SES.

Die Knoten einer SES können allgemein in *Entität* und *Deskriptiv*, als Kurzform für deskriptiver Knoten, eingeteilt werden. Zeigler und Hammonds [186] definieren eine Entität abstrakt als ein Ding der realen oder imaginären Welt. In der M&S repräsentieren Entitäten

entweder ein gekoppeltes Modell oder eine Komponente gemäß Abschnitt 2.1.2. Ebenso kann eine Entität eine abstrakte oder generische Komponente darstellen. Deskriptive Knoten beschreiben Beziehungen zwischen einzelnen Entitäten. Weiterhin können alle Entitäten und einige deskriptive Knoten Attribute spezifizieren, die in geschweiften Klammern angegeben werden. Nachfolgend werden die einzelnen Knotentypen mit Bezug zur SES in Abbildung 13 erläutert. Dabei werden Begriffe wie Vater, Kind, Nachfolger, Vorgänger, Pfad, Teilbaum etc. analog der Graphentheorie in [154, 101] verwendet und hier nicht näher erläutert.

Wurzel-Entität: ***Prozesskette***

Die Wurzel-Entität ist ein ausgezeichneter Knoten, welcher ausschließlich Nachfolger und keine Vorgänger besitzt. Er charakterisiert das modular-hierarchische Simulationsmodell als ein gekoppeltes Modell, dessen unterschiedliche strukturelle Ausprägungen hierarchisch durch die Nachfolger in der SES spezifiziert sind.

Blatt-Entitäten: ***Generator, FIFO, LIFO, Senke, ALD, ARS, VH, IH***

Blatt-Entitäten entsprechen, bis auf eine Ausnahme, den in der Modellbibliothek (MB) organisierten Komponenten. Die Zuordnung der Blatt-Entitäten zu den jeweiligen Komponenten erfolgt direkt über den Namen der Entität. Vor diesem Hintergrund muss für jedes Blatt eine entsprechende Komponente in der MB vorhanden sein. Auf diese Weise wird die in Abbildung 12 gezeigte formale Kopplung zwischen der SES und der MB realisiert. Die Attribute der Blatt-Entitäten repräsentieren die konfigurierbaren Parameter der zugehörigen Komponente in der MB. Beispielsweise definiert die Blatt-Entität *ALD* die Attribute Schnitttiefe *ap*, Vorschub *f* und Schnittgeschwindigkeit *vc* analog der zugehörigen Komponente in der MB in Abbildung 13.

Zeigler und Hammonds [186] definieren eine charakteristische Blatt-Entität *NONE*, die komplett anders behandelt wird und damit eine Ausnahme darstellt. Diese Blatt-Entität spezifiziert das *Nichts* und repräsentiert weder eine Modellkomponente noch besitzt diese Blatt-Entität Attribute. Der *NONE* Knoten ist in der SES in Abbildung 13 nicht enthalten. Er wird im nächsten Abschnitt wieder aufgegriffen.

Innere-Entitäten: ***APK, ALDs, Puffer, Härten***

Die inneren Entitäten besitzen immer einen Vater sowie mindestens ein Kind und können entweder gekoppelte Modelle oder abstrakte Entitäten darstellen. Entitäten gekoppelter Modelle besitzen ein Kind vom Typ Aspekt oder Multi-Aspekt. Abstrakte Entitäten besitzen im Rahmen der M&S in der Regel ein Kind vom Typ Spezialisierung. In der Abbildung 13 entsprechen die Entitäten *APK* sowie *ALDs* gekoppelten Modellen und *Puffer* und *Härten* abstrakten Entitäten.

Aspekt: ***Prozesskette-Dec, APK1-Dec, APK2-Dec***

Der deskriptive Aspekt-Knoten stellt im Rahmen der M&S die Zerlegung beziehungsweise eine Dekomposition einer Entität dar. Alle Knoten vom Typ Aspekt werden mit dem Suffix *Dec* (*engl. decomposition*) und einer einfachen Kante gemäß Abbildung 14 gekennzeichnet. Im Beispiel der Abbildung 13 beschreibt *Prozesskette-Dec* eine hierarchische Zerlegung der Entität *Prozesskette* in die Entitäten *Generator*, *FIFO*, *APK* und

Senke. Wie zum Beispiel an der Entität *APK* in Abbildung 13 gezeigt wird, kann eine Entität mehrere Aspekt-Knoten als Kinder besitzen. In einem solchen Fall kann eine Entität in unterschiedliche Entitäten zerlegt werden. Beispielsweise kann *APK* entweder unter dem Aspekt *APK1-Dec* oder *APK2-Dec* zerlegt werden. Eine solche Konstellation entspricht gemäß Abschnitt 2.3 einem Variationspunkt, der Auswahlmöglichkeiten innerhalb einer SES kennzeichnet. Die Auflösung solcher Variationspunkte erfolgt in der Phase der Variantengenerierung durch die Methode *pruning*.

Als Attribute weisen Aspekt-Knoten ausschließlich Kopplungsrelationen auf. Diese werden in geschweiften Klammern angegeben und spezifizieren die Kopplungen zwischen dem Vater und den Kindern des Aspekt-Knotens. Beispiele zur Spezifikation von Kopplungen sind in Abbildung 13 an *Prozesskette-Dec*, *APK1-Dec* und *APK2-Dec* gegeben. Dabei wird ein Tupel, wie zum Beispiel *(Generator.out, FIFO.in)*, wie folgt interpretiert:

E*s gibt eine Verknüpfung vom Port* ***out*** *der Entität* ***Generator*** *nach Port* ***in*** *der Entität* ***FIFO***.

Die Präpositionen *vom* und *nach* geben den Wirkzusammenhang im Sinne der Kopplungsrichtung wieder. Zusätzlich gilt die Gleichung 10.

$$(Generator.out, FIFO.in) \neq (FIFO.in, Generator.out) \tag{10}$$

Multi-Aspekt: ***ALD-MAsp***

Der Multi-Aspekt entspricht einem Spezialfall des Aspekt-Knotens. Er kennzeichnet die Zerlegung einer Entität in mehrere typengleiche Kind-Entitäten im Sinne einer mehrfachen Instanziierung und wird in der SES durch eine dreifache Kante sowie dem Suffix *MAsp* dargestellt. Der Multi-Aspekt kann zwei unterschiedliche Attribute definieren. Einerseits muss ein Wertebereich für die minimal und maximal mögliche Instanziierung typengleicher Kinder angegeben werden. Andererseits können analog dem Aspekt-Knoten Kopplungsrelationen spezifiziert werden. Demnach bildet der Multi-Aspekt einen weiteren Typ von Variationspunkt. Die Auflösung wird beim Pruning im nächsten Abschnitt diskutiert. In der Abbildung 13 kennzeichnet der Knoten *ALD-MAsp* die Zerlegung der Entität *ALDs* in mindestens eine und maximal zwei *ALD* Entitäten. Die Spezifikation der Kopplungen am Multi-Aspekt Knoten erfolgt analog zum Aspekt Knoten. Im Abschnitt 3.2.3 werden die Kopplungsbeziehungen für diesen Knotentyp näher betrachtet.

Spezialisierung: ***Puffer-Spec, Härten-Spec***

Knoten vom Typ Spezialisierung werden in der SES Syntax mit einer doppelten Kante und dem Suffix *Spec* dargestellt. Ein Spezialisierungs-Knoten kennzeichnet die Taxonomiebeziehung zwischen einem Vater und seinen Kindern. Mit Hilfe der Spezialisierung ist ves möglich, Klassifizierungen innerhalb einer SES zu spezifizieren. In der Abbildung 13 werden *FIFO* und *LIFO* der Kategorie *Puffer* zugeordnet, *IH* und *VH* der Kategorie *Härten*. Die Entitäten *Puffer* und *Härten* sind abstrakte Entitäten.

Die Spezialisierung bildet den dritten Typ eines Variationspunktes, bei dem zur Ableitung einer konkreten Struktur eine Auswahl getroffen werden muss.

Zur Modellierung mit einer SES müssen neben den diskutierten Knotentypen auch Grundannahmen beziehungsweise Axiome beachtetet werden. Die Axiome schränken den Interpretationsspielraum und Spezifikationsmöglichkeiten einer SES erheblich ein. Dadurch wird die Semantik einer SES konkretisiert und darüber hinaus die Gefahr der Fehlinterpretation oder Fehlspezifikation einer SES reduziert. Für eine valide SES müssen die folgenden sechs Axiome eingehalten werden.

AX1: ***Alternierender Modus***

Jeder Knoten muss vom Typ Entität oder Deskriptiv sein. Die Wurzel und die Blätter sind immer vom Typ Entität. Die Kinder eines Knotens vom Typ Entität müssen, soweit er kein Blatt ist, immer vom Typ Deskriptiv und die Kinder eines deskriptiven Knotens müssen immer vom Typ Entität sein.

AX2: ***Uniformität***

Zwei gleichnamige Knoten müssen vom selben Typ sein, die gleichen Attribute besitzen und isomorphe Teilbäume aufweisen.

AX3: ***Strikte Hierarchie***

Ein Knotenname darf nicht mehr als einmal innerhalb eines Pfades einer SES vorkommen.

AX4: ***Valide Brüder***

Die Kinder eines Knotens dürfen nicht den gleichen Namen besitzen.

AX5: ***Zugewiesene Variable***

Die Bezeichner aller Attribute eines Knotens müssen voneinander verschieden sein.

AX6: ***Vererbung***

Vater und Kinder einer Spezialisierung vereinen beim Pruning deren individuelle Namen, Attribute, Spezialisierungen und Aspekte.

Nach der Vorstellung der einzelnen Elemente und Axiome der SES werden im nächsten Unterabschnitt die zwei grundlegenden Methoden zur Ableitung einer konkreten Strukturvariante und zur Generierung eines modular-hierarchischen Simulationsmodells diskutiert.

3.2.2 Die Methoden *pruning* und *translation* zur Ableitung ausführbarer Simulationsmodelle

Die Spezifikation der unterschiedlichen Strukturen der Simulationsmodelle erfolgt beim SES/MB Framework in einer SES, wobei die Blätter der SES die zugehörigen Komponenten in der MB repräsentieren. Die Generierung in Form eines modular-hierarchischen Simulationsmodells erfolgt gemäß Abbildung 12 durch die Methoden *pruning* und *translation*. Erstere ermöglicht die Auflösung aller Variationspunkte innerhalb einer SES und führt zur Ableitung genau einer konkreten Strukturvariante. Tabelle 3 fasst alle Variationspunkte sowie Auswahlmöglichkeiten der SES in Abbildung 13 zusammen. Das Ergebnis einer *pruning* Operation ist eine *Pruned Entity Structure* (PES), welche genau eine Strukturvariante eines modular-hierarchischen Simulationsmodells beschreibt. Diese folgt

unmittelbar aus den Kopplungsrelationen, der Zuordnung von Blatt-Entitäten zu Komponenten in der MB und aus der Abbildung der Attribute der Blatt-Entitäten auf Parameter der Komponenten. An Hand dieser Informationen generiert die Methode *translation* unter Verwendung von Komponenten aus der MB ein ausführbares modular-hierarchisches Simulationsmodell.

Tabelle 3: Variationspunkte und qualitative sowie quantitative Auswahlmöglichkeiten der SES aus Abbildung 13.

Variationspunkte	**Auswahlmöglichkeiten**	
	qualitativ	*quantitativ*
APK	APK1-Dec, APK2-Dec	2
ALD-MAsp	ALD1, (ALD1, ALD2)	2
Puffer-Spec	FIFO, LIFO	2
Härten-Spec	VH, IH	2

Der beschriebene Vorgang zur Ableitung einer PES und der Generierung eines modular-hierarchischen Simulationsmodells soll am Beispiel der SES aus Abbildung 13 illustriert werden.

Gemäß Abbildung 13 und der Tabelle 3 besitzt die SES vier Variationspunkte mit jeweils zwei Auswahlmöglichkeiten. Insgesamt sind in der SES 9 unterschiedliche Strukturvarianten kodiert und somit können 9 unterschiedliche Simulationsmodelle generiert werden. Zur Ableitung einer PES muss zunächst in jedem Variationspunkt eine Auswahl getroffen werden. Das anschließende *pruning schneidet* an Hand der getroffenen Auswahl alle nicht gewählten Knoten eines Variationspunktes samt deren Nachfolger innerhalb einer SES weg. Tabelle 4 zeigt beispielhaft eine konkrete Auswahl für jeden Variationspunkt.

Tabelle 4: Auswahl einer konkreten Variante in jedem Variationspunkt gemäß Tabelle 3.

Variationspunkte	APK	ALD-MAsp	Puffer-Spec	Härten-Spec
Auswahl	APK1-Dec	1	LIFO	VH

Die resultierende PES ist in der Abbildung 15 zu sehen. Durch die Auswahl vom *APK1-Dec* im Variationspunkt *APK* wurden der Knoten *APK2-Dec* und alle seine Nachfolger entfernt. Damit entfällt die Auflösung der anderen Variationspunkte, die in der Tabelle 4 aufgelistet sind. Durch ein solches Vorgehen wird die hervorgehobene Problematik bezüglich des *toten Codes* aus Unterabschnitt 2.3.1 sowie 2.3.2 umgangen. Abschließend generiert die Methode *translation*, wie in Abbildung 15 gezeigt, ein modular-hierarchisches Simulationsmodell, welches durch einen zugehörigen Simulator ausgeführt und damit zum Experimentieren gemäß Abschnitt 2.2 verwendet werden kann.

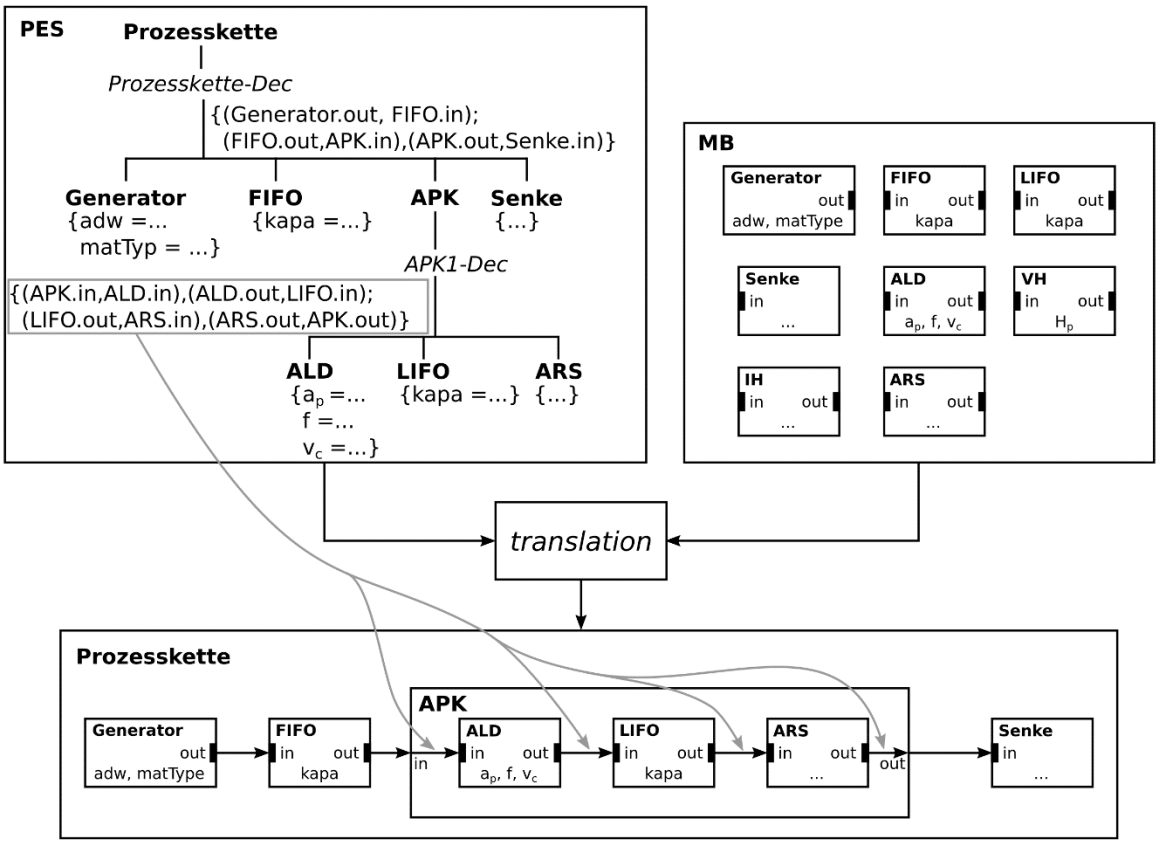


Abbildung 15: Darstellung einer PES, abgeleitet aus der SES in Abbildung 13, einer MB und der Generierung eines modular-hierarchischen Modells durch die Methode translation.

Allerdings besitzt das klassische SES/MB Framework einige Schwächen beziehungsweise offene Probleme, die in der Literatur nicht diskutiert werden. Diese sollen im weiteren Verlauf zunächst genannt und dann Lösungsvorschläge im Abschnitt 3.3 präsentiert werden.

3.2.3 Probleme des SES/MB Frameworks

Die offenen Probleme des SES/MB Frameworks werden am Beispiel der PES in Abbildung 16 diskutiert. Diese PES entsteht durch Pruning der SES in Abbildung 13, wenn die Auswahl an den Variationspunkten gemäß Tabelle 5 erfolgt.

Tabelle 5: Auswahl einer weiteren Variante in jedem Variationspunkt gemäß Tabelle 3.

Variationspunkte	APK	ALD-MAsp	Puffer-Spec	Härten-Spec
Auswahl	APK2-Dec	2	FIFO	VH

Durch die Auswahl von *APK2-Dec* in *APK* wird der *APK1-Dec* Knoten aus der SES entfernt. Die Auswahl *2* im *ALD-MAsp* bedeutet, dass das Kind *ALD* des Knotens *ALD-MAsp* zweimal instanziiert wird. Weiterhin müssen die beiden generierten Knoten an Hand des Axioms *Valide Brüder (AX4)* in *ALD1* und *ALD2* umbenannt werden. Infolge der Auswahl der Blatt-Entität *FIFO* im Variationspunk *Puffer-Spec* und durch das Anwenden des Axioms *Vererbung (AX6)* entsteht eine neue Blatt-Entität *FIFO_Puffer*. Nach demselben Muster entsteht auch die Blatt-Entität *VH_Härten*.

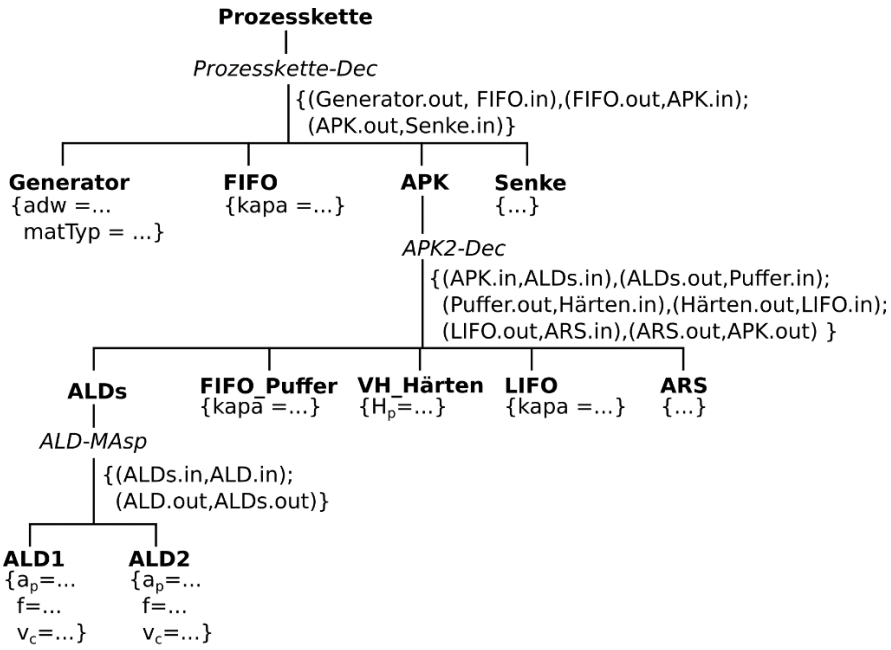


Abbildung 16: Eine weitere PES, abgeleitet aus der SES in Abbildung 13.

Nach der klassischen Theorie kann aus dieser PES und der zugehörigen MB kein ausführbares Simulationsmodell generiert werden. Der Hauptgrund liegt in der Anwendung der Axiome *AX4* und *AX6* sowie der Semantik des Multi-Aspekts. Dadurch werden die Namen der Blatt-Entitäten modifiziert und es entstehen neue Blatt-Entitäten, welche in der originären SES in Abbildung 13 nicht enthalten sind, wie beispielsweise die Entitäten *ALD1*, *ALD2*, *FIFO_Puffer* und *VH_Härten*. Diese neuen Blatt-Entitäten können nicht mehr eindeutig den Komponenten der MB zugeordnet werden. Des Weiteren stimmen die Kopplungsrelationen am Aspekt *APK2-Dec*, wegen der beschriebenen Modifikation der Entitäten, nicht mehr in Bezug auf die Kinder. Das Gleiche gilt für die Spezifikation der Kopplungsrelationen im Attribut des Knotens *ALD-MAsp*.

Ein weiteres offenes Problem stellt das Fehlen von konkreten Algorithmen für die Methoden *pruning* und *translation* dar. In der analysierten Literatur werden die Methoden nur abstrakt, wie im vorhergehenden Abschnitt, beschrieben und nicht konkretisiert. Ausschließlich die Arbeiten von Rozenblit und Zeigler [128] sowie Zeigler [185] zeigen einen stark vereinfachten Pseudocode für die Methode *pruning*, der die Auflösung von Spezialisierungen und Mulit-Aspekten aber nicht beinhaltet. Aus diesem Grund müssen entsprechende Algorithmen für beide Methoden entwickelt werden. In Anbetracht der analysierten Probleme werden im nächsten Abschnitt einige Lösungsvorschläge und Erweiterungen zum SES/MB Framework diskutiert.

3.3 Erweiterungen des SES/MB Frameworks

Die in diesem Abschnitt beschriebenen Erweiterungen und Lösungsvorschläge basieren maßgeblich auf den Arbeiten von Schmidt und Pawletta [143, 141], Schwatinski und Pawletta [148] sowie Pawletta et al. [114]. Alle eingeführten Erweiterungen werden an Hand der SES in Abbildung 17, welche eine restrukturierte SES bezüglich Abbildung 13 darstellt, diskutiert. Die zugehörige MB entspricht der MB aus Abbildung 13 und Abbildung 15.

Die SES in Abbildung 17 beinhaltet zwei *NONE* Knoten. Wie im vorherigen Abschnitt beschrieben repräsentieren diese Blatt-Entitäten das *Nichts*. In der gezeigten Kombination mit einer Spezialisierung ermöglicht der *NONE* Knoten das *Nichtauswählen* des Vaters einer Spezialisierung. Wird beispielsweise in der Spezialisierung *Härten-Spec* die Entität *NONE* ausgewählt, so bedeutet dies, dass keine Komponente *Härten* im zu generierenden modular-hierarchischen Simulationsmodell enthalten ist. Näheres hierzu wird im Abschnitt 3.3.5 mit Bezug zum Angang E behandelt.

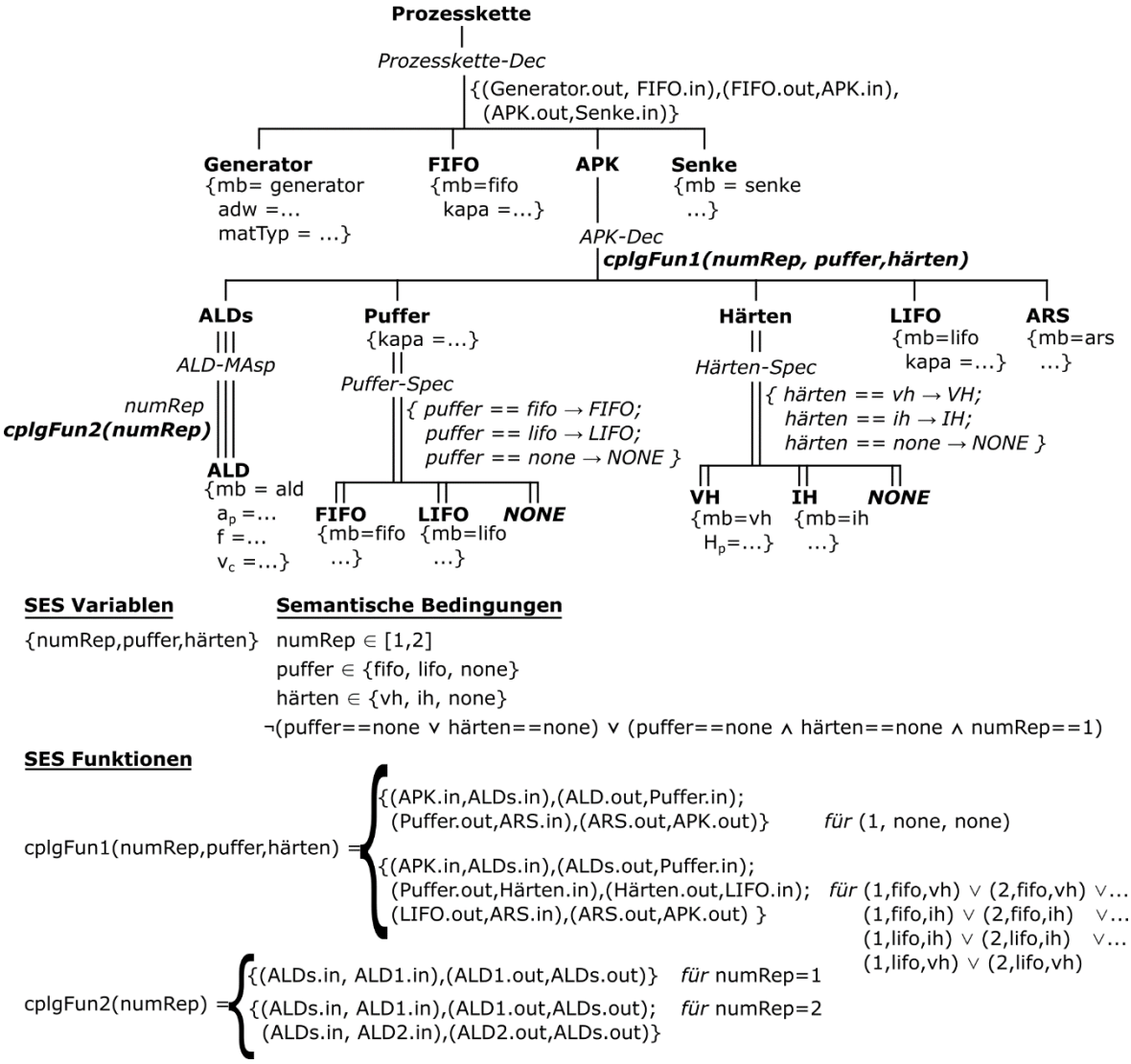


Abbildung 17: Restrukturierte SES mit maßgeblichen Erweiterungen bezüglich der SES in Abbildung 13.

Die wesentlichen Erweiterungen bilden (i) das charakteristische Attribut *mb* an den Blatt-Entitäten, (ii) zusätzliche Attribute mit Auswahlregeln, (iii) das Konzept der SES-Variablen, (iv) die Einführung der semantischen Bedingungen sowie (v) der SES-Funktionen. Tabelle 6 fasst alle Variationspunkte und Auswahlmöglichkeiten der SES aus Abbildung 17 zusammen. Wie in der Tabelle 6 zu erkennen ist, bilden die SES-Funktionen einen weiteren Ansatz zur Spezifikation von Variationspunkten.

Tabelle 6: Übersicht der Variationspunkte und Auswahlmöglichkeiten der SES aus Abbildung 17.

Variationspunkte		**Auswahlmöglichkeiten**	
Knoten	*SES-Funktionen*	*qualitativ*	*quantitativ*
ALD-MAsp		ALD1, (ALD1, ALD2)	2
Puffer-Spec		FIFO, LIFO, NONE	3
Härten-Spec		VH, IH, NONE	3
	cplgFun1	vgl. Abbildung 17	2
	cplgFun2	vgl. Abbildung 17	2

Nachfolgend erfolgt eine detaillierte Betrachtung der eingeführten Erweiterungen auf Basis der SES in Abbildung 17.

3.3.1 Charakteristisches *mb* Attribut

Bezugnehmend auf Abschnitt 3.2.3 basieren die Hauptprobleme der originären SES auf der Modifikation der Entitäten-Namen durch Vererbung und die Erschaffung neuer Entitäten durch den Multi-Aspekt beim Pruning. Dadurch können Blatt-Entitäten nicht mehr eindeutig den Komponenten zugeordnet und somit kein ausführbares Simulationsmodell generiert werden. Das charakteristische *mb* Attribut wirkt diesem Problem entgegen. An jeder Entität, welche mit der MB korrespondiert, wird ein Attribut *mb* definiert. Der Wert des Attributs kodiert den Namen der referenzierten Modellkomponente in der MB. Wichtig ist darauf hinzuweisen, dass dieses Attribut *keinem* Parameter der Modellkomponente entspricht. Vor diesem Hintergrund besitzt die Entität *VH* in Abbildung 17 zwei Attribute *Hp =...* und *mb = vh*, wobei nur das Attribut *Hp* einem Parameter entspricht.

Wird beispielsweise beim Pruning *FIFO* an der Spezialisierung *Puffer-Spec* ausgewählt und durch das Axiom AX6 *zu FIFO_Puffer* kombiniert, so erbt die neue Entität alle Attribute der Entitäten *Puffer* und *FIFO*. Anders als in Abschnitt 3.2.3 beschrieben, referenziert jetzt die Entität *FIFO_Puffer* über das Attribut *mb=fifo* unabhängig vom Knotennamen eindeutig eine zugehörige Modellkomponente in der MB, wobei hier *case- insensitivity* angenommen wird. Durch ein solches Vorgehen bleibt in den beschriebenen Fällen die formale Kopplung zwischen der SES und der MB erhalten.

3.3.2 Konzept der SES-Variablen und Auswahlregeln

Auswahlregeln und SES-Variablen werden beim Pruning evaluiert. SES-Variablen stellen Einflussgrößen einer SES dar und besitzen den Aufbau gemäß Gleichung 1 im Unterabschnitt 2.1.2. Bezogen auf eine SES haben sie einen globalen Geltungsbereich und besitzen eine *lesende* Zugriffsbeschränkung. SES-Variablen werden meist in Kombination mit SES-Funktionen sowie Auswahlregeln eingesetzt. Sie können natürlich auch, wie in [141] gezeigt, zur flexiblen Parametrierung von Attributen einer Entität eingesetzt werden. Die SES in Abbildung 17 definiert die SES-Variablen *puffer*, *numRep* und *härten*. Vor dem *Pruning* müssen jeder SES-Variablen, durch den Anwender, Werte zugewiesen werden, so dass die Variationspunkte aufgelöst werden können. In diesem Sinne können die SES-Variablen als eine Nutzerschnittstelle verstanden werden.

Auswahlregeln spezifizieren und visualisieren, wie eine Auswahl an den Variationspunkten zu treffen ist. In der Abbildung 17 sind an den Spezialisierungen *Puffer-Spec* und *Härten-Spec* Auswahlregeln in Form von einer Menge logischer Ausdrücke beschrieben. Im Weiteren können Auswahlregeln in Kombination mit den SES-Variablen zur Spezifikation von baumübergreifenden Beziehungen (*engl. cross-tree relationships*) genutzt werden. Auf diese Weise können innerhalb einer SES zum Beispiel, analog dem Featuremodell in [75, 146], *require* oder *exclude* Beziehungen zwischen Entitäten, unbeachtet deren hierarchischer Anordnung in der SES, beschrieben werden. Die *require* Beziehung wurde im Abschnitt 2.3.2 in der Abbildung 9 gezeigt. Die *exclude* Beziehung entspricht dem Gegenteil von *require*. Allerdings hebt Wittern [180] hervor, dass baumübergreifende Beziehungen Raum für Inkonsistenzen oder Konflikten bieten. Besitzt also eine SES baumübergreifende Beziehungen zwischen mehreren Entitäten und wird die SES zu einem späteren Zeitpunkt modifiziert, so müssen alle baumübergreifenden Beziehungen auf ihre Gültigkeit überprüft werden. Andernfalls kann die *pruning* Operation zu einer fehlerhaften PES führen.

3.3.3 SES-Funktionen

Mit SES-Funktionen werden Abhängigkeiten von den aktuellen Belegungen der SES-Variablen spezifiziert. Ihre Auswertung erfolgt beim Pruning. SES-Funktionen werden als Variationspunkt in Attributen der deskriptiven Knoten oder Entitätenknoten oder zur Definition attributübergreifender Beziehungen (*engl. cross-attribute relationships*) genutzt. Die Verwendung von SES-Funktionen als Variationspunkt ermöglicht oft eine Zusammenfassung von Teilbäumen in einer SES und reduziert so die Komplexität des Baumes. Dieses Prinzip ist in Abbildung 17 gezeigt. Bei Betrachtung des Variationspunktes *APK* in der originären SES in Abbildung 13 erkennt man, dass der Teilbaum *APK1-Dec* samt der Menge der Kopplungen einer Teilmenge des Teilbaumes von *APK2-Dec* entspricht. Die SES-Funktion *cplgFun1(numRep, puffer, härten)* in der SES in Abbildung 17 ermöglicht die Vereinigung beider Knoten im Aspekt *APK-Dec* und reduziert auf diese Weise die Komplexität des Baumes. Ein weiteres schematisches Beispiel befindet sich im Anhang B.

Ein weiteres Beispiel für den Einsatz von SES-Funktionen ist die Spezifikation alternativer Kopplungsbeziehungen, wie in der SES in Abbildung 17 am Knoten *ALD-MAsp* gezeigt. Mit der SES-Funktion *cplgFun2(numRep)* werden in Abhängigkeit von der aktuellen Wertebelegung der SES-Variablen *numRep* die Kopplungsbeziehungen für unterschiedlich viele Replikationen des Multi-Aspekts spezifiziert. Somit bieten SES-Funktionen einen Lösungsansatz für das in Abschnitt 3.2.3 diskutierte Problem unvollständig spezifizierter Kopplungen. In der gleichen Weise können SES-Funktionen als Variationspunkt zur flexiblen Parametrierung von Attributen an Entitätenknoten eingesetzt werden, wie zum Beispiel in Schmidt und Pawletta [141] und in Pawletta et al. [114] gezeigt.

Wie bereits erwähnt, können SES-Funktionen zur Spezifikation von attributübergreifenden Beziehungen genutzt werden. Damit können analog zu baumübergreifenden Beziehungen *require* oder *exclude* Beziehungen über Attribute von Entitäten, unabhängig von deren hierarchischer Anordnung in einer SES, realisiert werden.
Allerdings birgt eine inflationäre Verwendung von SES-Funktionen auch die Gefahr der Unübersichtlichkeit. Damit können SES-Funktionen auch zu einer potentiellen Fehlerquelle werden. Deshalb ist ihr Einsatz genau abzuwägen. Nicht in jedem Fall führt eine Komplexitätsreduktion des Baumes durch SES-Funktionen zu mehr Übersichtlichkeit. Allgemein wird

durch die Einführung der SES-Funktionen eine gemischte Beschreibung aus textueller und graphischer Notation einer SES ermöglicht.

3.3.4 Semantische Bedingungen

Die semantischen Bedingungen konkretisieren den zulässigen Wertebereich der SES-Variablen. Weiterhin können mit den semantischen Bedingungen logische Ausdrücke zur Einschränkung der Kombinationen von Wertebelegungen der SES-Variablen spezifiziert werden. Das heißt, durch die Einschränkung der Kombinationen von SES-Variablenbelegungen kann die Anzahl der ableitbaren PES und damit die Anzahl der möglichen Strukturvarianten eingeschränkt werden. Beispielsweise kodiert die SES in Abbildung 13 neun unterschiedliche Strukturvarianten. Durch die Restrukturierung der SES in Abbildung 17 sind nun aber 18 Strukturen spezifiziert. Die letzte Zeile der *semantischen Bedingungen* in Abbildung 17 definiert eine Restriktion, mit der unzulässige SES-Variablen-Kombinationen ausgeschlossen werden, sodass die Anzahl der gültigen Strukturvarianten wieder auf die ursprünglichen neun Varianten begrenzt wird. Nur wenn alle semantischen Bedingungen *wahr* sind, kann eine gültige PES abgeleitet werden. Beispielsweise führt eine Belegung der SES-Variablen mit:

- *numRep = 1, puffer = none, härten = none*; zu einer gültigen PES und
- *numRep = 2, puffer = none, härten = none*; zu einer ungültigen PES.

3.3.5 Die Methoden *pruning* und *translation* im Kontext der Erweiterung

Die eingeführten Modifikationen lösen die zuvor beschriebenen Probleme der originären SES-Definition, die speziell beim Ableiten einer PES auftreten kann. Aufgrund der Modifikationen müssen jetzt natürlich die Methoden *pruning* und *translation* zur Generierung einer konkreten Variante eines ausführbaren Simulationsmodells angepasst oder neu entwickelt werden. In der neueren Literatur zum Thema SES oder SES/MB Framework werden keine Algorithmen für die Methoden angegeben und die Algorithmen in den älteren Quellen, sind auf Grund kontinuierlicher Erweiterungen der SES als obsolet zu betrachten. Aus diesem Grund mussten für beide Methoden neue Algorithmen entwickelt werden. Im Anhang C und D sind diese als abstrakte Algorithmen aufgezeigt. Formal setzen die beiden Algorithmen die Transformationsmethoden in Gleichung 11 und 12 um.

$$pruning: SES \times SESVariables \rightarrow PES \tag{11}$$

$$translation: MB \times PES \rightarrow EM \tag{12}$$

Bezugnehmend auf die Abschnitte 3.1 und 3.2 entspricht eine SES im Sinne der Graphentheorie einem Baum. Demzufolge ist der *pruning* Algorithmus im Anhang C an eine rekursive Tiefensuche (*engl. Depth-First Search*) aus dem Teilgebiet der uninformierten Suche angelehnt. Ausgehend von den Wertezuweisungen der SES-Variablen, werden alle Variationspunkte einer SES aufgelöst. Zusätzlich werden die Kopplungsbeziehungen den strukturellen Änderungen angepasst, welche beim Pruning aus den Axiomen AX4 und AX6 resultieren. Der Algorithmus für die Methode *translation* ist ebenfalls rekursiv aufgebaut, entspricht aber *keiner* Tiefensuche. Die PES wird rekursiv durchlaufen und an Hand des

Entitätentyps (Blatt-Entität oder Innere-Entität) sowie den Kopplungsbeziehungen an den Aspekt-Knoten wird ein ausführbares Simulationsmodell unter Nutzung der MB generiert. Im Anhang E werden zwei Beispiele zur Ableitung einer PES und Generierung eines Simulationsmodells präsentiert. Die Einordnung des SES/MB Frameworks in das Variantenmanagement gemäß Abschnitt 2.3 wird im nächsten Abschnitt diskutiert.

3.4 Einordnung des SES/MB Frameworks in das Variantenmanagement

Die im Abschnitt 2.3 identifizierten Phasen des Variantenmanagements: (i) *Varianten-analyse*, (ii) *-formalisierung*, (iii) *-implementierung* und (iv) *-generierung*, sollen im Kontext des ASIM-Vorgehensmodells der M&S nach Abbildung 1 und des SES/MB Frameworks betrachtet werden. Abbildung 18 zeigt das entsprechend erweiterte Vorgehens-modell.

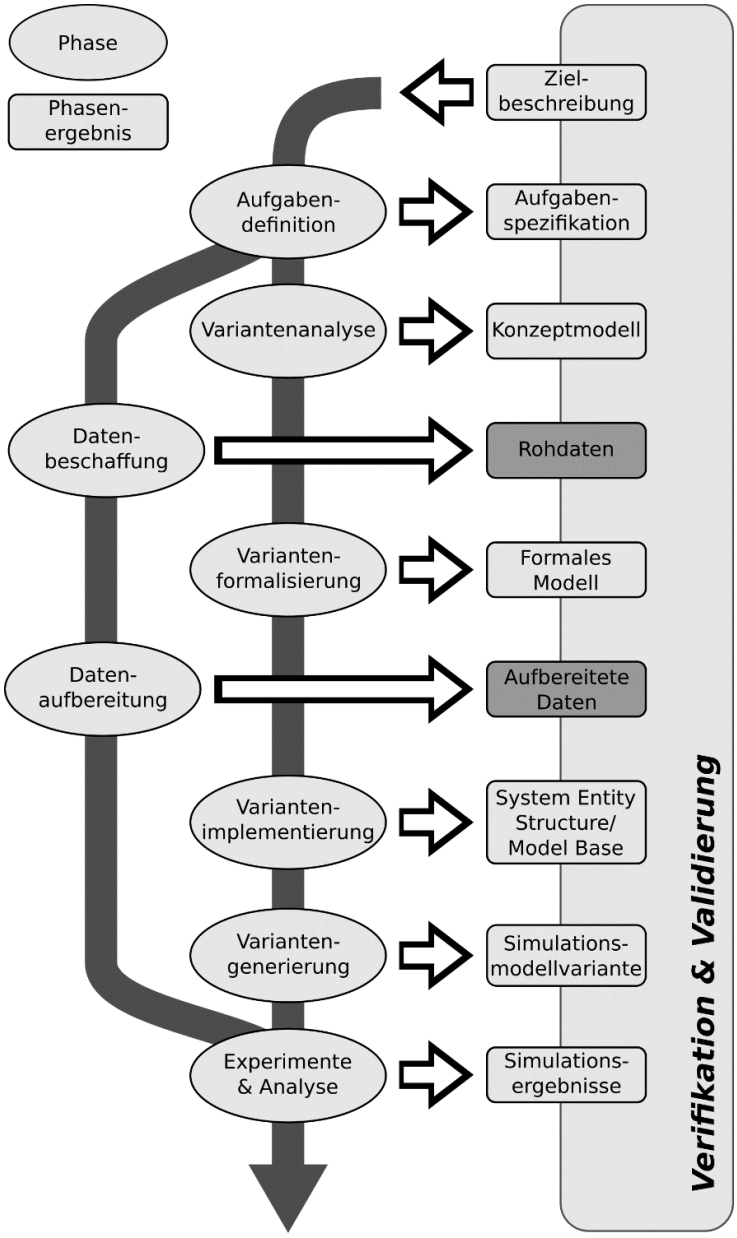


Abbildung 18: Erweitertes ASIM-Vorgehensmodell im Kontext des Variantenmanagements und des SES/MB Frameworks.

Die *Variantenanalyse* erfolgt ausgehend von einer zu betrachtenden Systemfamilie. Das übergeordnete Ziel ist hierbei die Schaffung eines holistischen Variantenverständnisses über alle Systemvarianten der Systemfamilie. Zunächst muss jede Systemvariante hinsichtlich der Systemstruktur und der dynamikbeschreibenden Komponenten analysiert werden. Im nächsten Schritt erfolgt eine Analyse aller Systemstrukturen hinsichtlich Gemeinsamkeiten, Unterschieden und Schnittmengen, unter Beachtung der formalen Kopplung mit den dynamikbeschreibenden Komponenten. Darüber hinaus sind Restriktionen der Variabilität zu identifizieren. Ist die Variabilität der Systemfamilie auf diese Weise identifiziert, so kann zur Spezifikation des Konzeptmodells bereits das SES/MB Framework eigensetzt werden. Mit einer SES wird die Variabilität in Form von Variationspunkten spezifiziert. Dies erfolgt unter Verwendung der unterschiedlichen Knotentypen und deren Attributen sowie mit SES-Variablen und SES-Funktionen. Insbesondere die Variabilität bezüglich der Kopplungsrelationen kann durch SES-Funktionen beschrieben werden. Die identifizierten Restriktionen werden als semantische Bedingungen definiert. Die wesentlichen dynamikbeschreibenden Komponenten können mit klassischen Methoden als Konzeptmodelle spezifiziert werden. Dabei sind die Definition der Ein- und Ausgabeschnittstellen und der konfigurierbaren Parameter für die formale Kopplung mit der SES wesentlich.

Bei der *Variantenformalisierung* kann auf den von Zeigler und Hammonds [186] entwickelten Formalismus zur formalen Spezifikation der SES zurückgegriffen werden. Zur Formalisierung der Komponenten der MB kann beispielsweise auf den DEVS Formalismus nach Tabelle 1 im Unterabschnitt 2.1.2 aufgesetzt werden. Jedoch ist, wie bereits im zweiten Kapitel diskutiert, die *strenge Formalisierung* in der Literatur umstritten.

Nach der M&S Theorie sind die beiden vorangestellten Phasen von der eingesetzten M&S-Softwareumgebung unabhängig. Erst während der *Variantenimplementierung* erfolgt eine konkrete Bindung an eine spezifische M&S-Softwareumgebung. Im Fall von MATLAB/Simulink kann die SES unter Nutzung der im Rahmen dieser Arbeit entwickelten SES-Toolbox für MATLAB/Simulink [117, 114] implementiert werden. Die MB ist in Form einer Simulink-Modellbibliothek zu implementieren.

Die *Variantengenerierung* erfolgt durch Wertzuweisungen an die SES-Variablen der zugehörigen SES und der anschließenden Anwendung der Methoden *pruning* und *translation*. Durch die Methode pruning wird anhand der SES genau eine Strukturvariante und Komponentenkonfiguration eines Simulationsmodells in Form der PES abgeleitet. Die Methode translation erstellt dann aus der PES und der zugehörigen MB eine ausführbare Variante eines Simulationsmodells. Ergänzend sei erwähnt, dass bei der Variantengenerierung auf Basis des SES/MB Frameworks auch eine Menge unterschiedlicher Simulationsmodellvarianten generiert werden kann.

Die unterschiedlichen Varianten ausführbarer Simulationsmodelle werden in der nachfolgenden Phase *Experimente & Analyse* in der spezifischen M&S-Umgebung sequentiell oder parallel/verteilt ausgeführt und die Simulations- und Experimentergebnisse werden evaluiert. Demgemäß kann das SES/MB Framework durchgehend von der Konzeptmodellierung bis zur Variantengenerierung eingesetzt werden.

3.5 Zusammenfassung und Diskussion

In diesem Kapitel wurde zunächst das klassische System Entity Structure/Model Base (SES/MB) Framework betrachtet. Beim SES/MB Framework erfolgt eine strenge Trennung bezüglich der Spezifikation der Systemstruktur und der Systemdynamik. Die Systemdynamik wird in konfigurierbaren Komponenten mit definierten Ein- und Ausgabeschnittstellen beschrieben, welche als wiederverwendbare Komponenten in einer MB organisiert werden. Mit einer SES werden eine Menge unterschiedlicher Systemstrukturen und Parameterkonfigurationen in Form eines Baumes beschrieben. Über die Blätter der SES wird eine formale Kopplung zwischen der SES und der MB realisiert. Die Variantengenerierung einer konkreten ausführbaren Simulationsmodellvariante erfolgt durch die Anwendung der Methoden *pruning* und *translation*.

Bei der Analyse des klassischen SES/MB Frameworks wurden bestehende Probleme identifiziert. Ein wesentliches Problem betrifft die formale Kopplung der SES und der MB, die dazu führt, dass nicht in jedem Fall eine ausführbare Variante eines Simulationsmodells generiert werden kann. Weiterhin fehlen konkrete Algorithmen für die Methoden *pruning* und *translation* bezüglich der kontinuierlichen Weiterentwicklungen der SES-Theorie.

Zur Überwindung der identifizierten Probleme wurden verschiedene Erweiterungen eingeführt: (i) das charakteristische Attribut *mb*, (ii) das Konzept der SES-Variablen, (iii) Auswahlregeln, (iv) semantische Bedingungen sowie (v) SES-Funktionen. Durch die Erweiterungen kann außerdem die Übersichtlichkeit einer SES verbessert und die Gefahr von Fehlspezifikationen reduziert werden. Die Ergebnisse sind in einem Softwareprototypen, der SES-Toolbox for MATLAB/Simulink [117, 114], implementiert worden. Weiterhin wurde der durchgehende Einsatz des SES/MB Frameworks im Kontext eines um das Variantenmanagement erweiterten ASIM-Vorgehensmodells diskutiert.

Die Betrachtung des SES/MB Frameworks erfolgte bisher ausschließlich im Kontext der Variantenanalyse, -implementierung und -generierung unterschiedlicher Simulationsmodelle. Das übergeordnete Ziel dieser Arbeit ist das Variantenmanagement von Simulationsexperimenten. Zur Spezifikation einer Menge unterschiedlicher Simulationsexperimentvarianten und deren automatischer Generierung muss zunächst eine allgemeine Struktur für Simulationsexperimente erarbeitet werden. Dabei soll auf dem Konzept des SES/MB Frameworks aufgebaut werden. Diese Problemstellung wird im nächsten Kapitel untersucht.

4 Variantenmanagement simulationsbasierter Experimente

In diesem Kapitel wird ein Variantenmanagement für Simulationsexperimente vorgestellt. Dazu wird zunächst ein allgemeiner Ansatz zur Strukturierung von Simulationsexperimenten vorgeschlagen. Dieser basiert auf der systematischen Trennung eines zu untersuchenden Systems vom Kontext einer spezifischen Problemstellung. Auf dieser Grundlage erfolgt ein Vorschlag zur modular-hierarchischen Beschreibung von Simulationsexperimenten. Der modular-hierarchische Aufbau von Experimenten bildet die Grundlage für das Variantenmanagement unter Verwendung des System Entity Structure/Model Base Frameworks. An einem vereinfachten Beispiel wird das Variantenmanagement von Experimenten präsentiert und diskutiert. Abschließend erfolgt eine Zusammenfassung.

4.1 Experimental Frame als Kontextspezifikation für ein Simulationsmodell

In Anlehnung an Traoré und Muzy [166] sind Simulationsmodelle für sich allein genommen bedeutungslos und nichtssagend. Als Grund hierfür nennen die Autoren die Abwesenheit des *Kontextes*. Kang et al. [74] zählen zum Kontext unter anderem die Systemumgebung. In der M&S folgt nach Traoré und Muzy [166] der Kontext eines Simulationsmodells aus dem Zusammenhang mit den *Zielstellungen*, *Randbedingung*en und weiteren *Annahmen*, welche sich auf eine zu untersuchende Problemstellung beziehen. Der Kontext eines Simulationsmodells bildet einen essentialen Bestandteil eines Simulationsexperiments und ist stets system- sowie zweckgebunden. Wie in Abbildung 19 gezeigt, kann der Kontext analog zur Modellbildung eines Systems in Form eines *Experimental Frames (EF)* modelliert werden. Demnach sollten zur Modellbildung eines EF die Grundlagen der Modellbildung gemäß Kapitel 2 herangezogen werden.

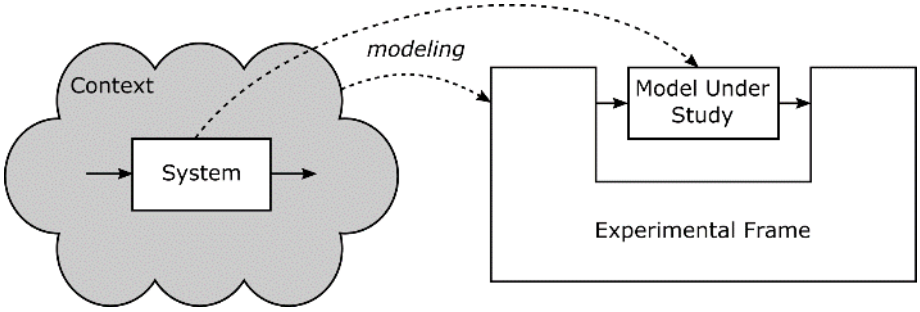


Abbildung 19: Beziehungen zwischen System und Kontext zu Modell und Experimental Frame, modifiziert aus [166].

Abweichend von der eingeführten Begriffsdefinition wird ab jetzt ein Modell eines Systems, gemäß Abbildung 19, als ein *Model Under Study (MUS)* bezeichnet und *nicht* wie bisher mit dem Simulationsmodell gleichgesetzt. Der Grund für diese Terminologie wird zu einem späteren Zeitpunkt in diesem Abschnitt diskutiert. Demnach konkretisiert der EF den Kontext eines MUS. Weiterhin ergänzen Foures et al. [43, 42], dass der EF ein vereinfachtes Modell einer konkreten Systemumgebung repräsentieren kann, was sich mit den Ansichten von Kang et al. [74] deckt. Demnach kann allgemein ein Versuchsobjekt (System) als ein MUS und die Versuchsumgebung (Systemumgebung) abstrahiert als ein EF modelliert

werden. Das Simulationsmodell als Versuchsaufbau ist dann eine Kombination aus MUS und EF.

Die grundlegenden Ideen zum EF gehen auf Zeigler [82] zurück und wurden seither in unterschiedlichen Arbeiten verwendet und weiterentwickelt. Schmidt et al. [139] zeigen die Verwendung des EF zur Spezifikation eines Testmodells im Rahmen der Testautomatisierung. Kim et al. zeigen in [77], wie der EF bei der parallelen und verteilten M&S unter Verwendung von *Hadoop* und *MapReduce* eingesetzt werden kann. Nader et al. [104] implementieren einen allgemeinen EF zum Speichern und Vergleichen von Simulationsergebnissen im Rahmen simulationsbasierter Experimenten zur Ausbreitung von Waldbränden. Röhl et al. [126] entwickelten auf Basis des EF einen flexiblen Ansatz zur Bewertung unterschiedlicher Strategien von *Diensten in Ad-hoc Netzen* in der M&S-Umgebung James II.

Die prinzipielle softwaretechnische Umsetzung des EF wird im nächsten Unterabschnitt aufgezeigt.

4.1.1 Softwaretechnische Umsetzung des Experimental Frames

Das Experimental Frame (EF) wird als ein gekoppeltes Modell auf einer Ebene mit dem Model Under Study (MUS) *im* Simulationsmodell umgesetzt. Demnach interagiert der EF *zur* Simulationslaufzeit mit dem MUS. Auf diese Weise besteht das Simulationsmodell, anders als in den Kapiteln zuvor, jetzt aus einem Modell eines Systems, dem MUS, und einem Modell eines Kontextes, dem EF. Gemäß Zeigler [185] sowie Traoré und Muzy [166] besteht ein EF aus den Komponenten (i) *Generator*, (ii) *Transducer* und (iii) *Acceptor*. Demnach entspricht das EF einem modular-hierarchischen Modell. Dieser Sachverhalt wird in Abbildung 20 gezeigt. Vollständigkeitshalber ist in Abbildung 20 der zur Ausführung einer Simulation notwendige Simulator mit aufgenommen.

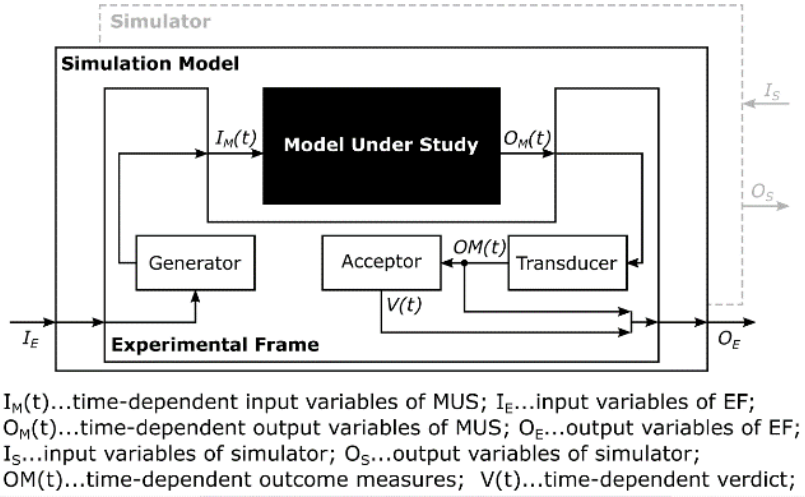


Abbildung 20: Schematisches Blockschaltbild eines Simulationsmodells mit EF und MUS sowie ausführendem Simulator.

Die in der Abbildung 20 gezeigten Kopplungsrelationen zwischen den einzelnen Komponenten des EF sind nur beispielhaft angedeutet. Diese können sich in Abhängigkeit einer zu lösenden Problemstellung ändern. Die Interaktion zwischen dem MUS und dem EF erfolgt

ausschließlich über die wohldefinierten Ein-/Ausgabeschnittstellen, welche die Einflussgrößen und Ausgabegrößen abbilden. Im Gegensatz zur Beschreibung im Abschnitt 2.1.1 werden in Kombination mit dem EF gemäß Zeigler [185] sowie Traoré und Muzy [166] die Einflussgrößen mit *I* und die Ausgabegrößen mit *O* gekennzeichnet. Weiterhin unterscheidet man die Einfluss- und Ausgabegrößen bezogen auf das EF und MUS. Erstere sind in der Abbildung 20 mit I_E und O_E bezeichnet, während die Einfluss- und Ausgabegrößen bezogen auf das MUS zeitbezogen mit $I_M(t)$ sowie $O_M(t)$ angegeben sind. Das bedeutet, dass letztere Größen während des Simulationslaufes zeitlichen Änderungen unterliegen. Demnach können $I_M(t)$ sowie $O_M(t)$ mit Signalen oder Trajektorien assoziiert werden. Wie bereits im Abschnitt 2.1.2 beschrieben, werden die Einflussgrößen und Ausgabegrößen des Simulators in der Literatur oft nicht betrachtet, was strenggenommen nicht korrekt ist. Zur vollständigen Korrektheit sind in der Abbildung 20 die Einflussgrößen und Ausgabegrößen bezogen auf einen Simulator mit I_S und O_S angegeben. Anschließend werden die drei Komponenten des EF (i) *Generator*, (ii) *Transducer* und (iii) *Acceptor* näher beschrieben. Darüber hinaus werden Parallelen zu den im Abschnitt 2.2.2 beschriebenen und den dort betrachteten Aufgaben diskutiert.

Generator

In der einschlägigen Literatur, wie zum Beispiel in [187, 185, 78], wird die Komponente *Generator* als eine *aktive* Komponente ohne Eingabeschnittstelle beschrieben. Das Wort *aktiv* weist darauf hin, dass der Generator selbstständig Zeitereignisse einplant und erst nach dem Eintreten eines Zeitereignisses eine Ausgabe generiert. Es wird jedoch nicht definiert, warum ein Generator keine Eingangsschnittstellen besitzen darf. Somit wird, wie in Abbildung 20 gezeigt, sich an der Literatur [102, 166, 51] orientiert, die eine Menge von Eingangsschnittstellen für einen Generator zulassen. Die Ausgaben eines Generators können periodische oder aperiodische Trajektorien beziehungsweise Signale sein. Wie in Abbildung 20 schematisch gezeigt, kann ein Generator zum Beispiel Einflussgrößen I_E entgegennehmen, diese in zeitbezogene Einflussgrößen $I_M(t)$ umrechnen und dem MUS zuführen. Mit Bezug auf den Abschnitt 2.2.1 übernimmt der Generator somit die Aufgabe *Konfiguration*, womit speziell die Konfiguration der Einflussgrößen $I_M(t)$ gemeint ist.

Transducer

Der Transducer sammelt während eines Simulationslaufes die zeitbehafteten Ausgabegrößen $O_M(t)$ des MUS und verdichtet diese zu Zielgrößen, bezogen auf die Zielstellung einer zu untersuchenden Problemstellung. Beispielsweise kann ein MUS eine Prozesskette repräsentieren, die als $O_M(t)$ die zeitliche Lastaufnahme $P(t)$ berechnet. Als eine zentrale Zielstellung sind zum Beispiel, auf Basis von $P(t)$ die aufgenommene elektrische Arbeit $W_E(t)$ und die Energiekosten $K_E(t)$ der Prozesskette zu berechnen. Somit würde der Transducer über seine Eingangsschnittstelle $P(t)$ aufnehmen und auf Basis seiner Dynamikbeschreibung die gewünschten Zielgrößen $OM(t) = \{W_E(t), K_E(t)\}$ berechnen. Auf die gleiche Art und Weise kann der Transducer im Rahmen einer simulationsbasierten Optimierung die charakteristische Zielfunktion codieren. Demnach würde die Zielfunktion Bestandteil des Simulationsmodells sein, was gemäß Unterabschnitt 2.2.2 dem Vorschlag

von Law und Kelton [91], Deckert [28] sowie Fu [46] entspricht. Allgemein gesehen übernimmt der Transducer mit Bezug auf Abschnitt 2.2.2 die Aufgaben *Datenakquisition* und *Datenanalyse*.

Acceptor

Der Acceptor spezifiziert Randbedingungen in Form von Toleranzen und Grenzwerten für einen Simulationslauf und überwacht deren Einhaltung zur Simulationslaufzeit. Hierzu übermitteln die anderen Komponenten über Kopplungsbeziehungen entsprechende Werte. Beispielsweise übermittelt in der Abbildung 20 der Transducer die *OM(t)* an den Acceptor und dieser überprüft die Einhaltung der vordefinierten Randbedingungen. Gleichzeitig wird ein Urteil (*engl. verdict*) über die zeitliche *Einhaltung* oder *Nichteinhaltung* der Randbedingungen als binärer Wert über die Ausgabeschnittstelle *V(t)* ausgegeben. Natürlich können auch Fuzzy-Mengen zur Kategorisierung eines Urteils verwendet werden. Bei Verletzung von kritischen Randbedingungen oder beim Erreichen der gewünschten Ergebnisse ermöglicht der Acceptor beispielsweise einen zustandsbasierten Simulationsabbruch.

Der Acceptor übernimmt nach Abschnitt 2.2.2 die Aufgabe *Bewertung*. Neu hinzu kommt die Aufgabe des zustandsbasierten Simulationsabbruchs, die nachfolgend als *Terminierung* bezeichnet wird. Nach Kim [78] findet der Acceptor speziell beim Durchführen von *Steady-State-Analysen* oder in Analogie zu Schmidt et al. [139] beim Testing seinen Einsatz.

Selbstverständlich können auch einzelne Komponenten des EF, wenn sie zur Problemlösung nicht benötigt werden, auch weggelassen werden. Nach einem Simulationslauf können die zeitorientierten Zielgrößen *OM(t)* und das zugehörige Urteil *V(t)* als Ausgabegrößen O_E des EF zurückgegeben werden. Die Anwendung des EF wird im nächsten Unterabschnitt anhand eines einfachen Beispiels demonstriert.

4.1.2 Beispiel zum Experimental Frame

Abbildung 21a zeigt schematisch ein MUS, verknüpft mit einem modular-hierarchisch aufgebauten EF. Das MUS ist eine Prozesskette gemäß Kapitel 3 mit nur einer Prozessoperation CNC-Drehen. Der interne Aufbau des MUS spielt nur eine untergeordnete Rolle. Über die Eingangsschnittstellen erhält das MUS zur Simulationslaufzeit zeitabhängig Rohlinge *in(t)* und einen zugehörigen G-Code *gCode(t)*. Das MUS berechnet während des Simulationslaufes ein zeitbezogenes Lastprofil *P(t)* und gibt dieses sowie die Fertigteile *out(t*) über die Ausgabeschnittstellen an den EF aus. Gemäß Abbildung 20 entsprechen die Einflussgrößen $I_M(t)$ des MUS der Eingangsschnittstelle *{gCode(t), in(t)}* und die Ausgabegrößen $O_M(t)$ der Ausgabeschnittstelle *{P(t), out(t)}*. Die $I_M(t)$ werden durch den Generator aus dessen Einflussgrößen I_E = *{ap, f, vc}* abgeleitet. Die Berechnung der gewünschten Zielgrößen Lastspitze (*loadPk*) und Durchsatz (*thrput*) erfolgt im Transducer. Dieser nimmt die $O_M(t)$ des MUS entgegen, berechnet daraus die gewünschten Zielgrößen *OM={loadPk, thrput}* und gibt diese direkt als Ausgabegrößen O_E des EF nach Beendigung des Simulationslaufes zurück. In diesem Fall entsprechen die vom Transducer berechneten Zielgrößen *OM* unmittelbar den Ausgabegrößen O_E des EF.

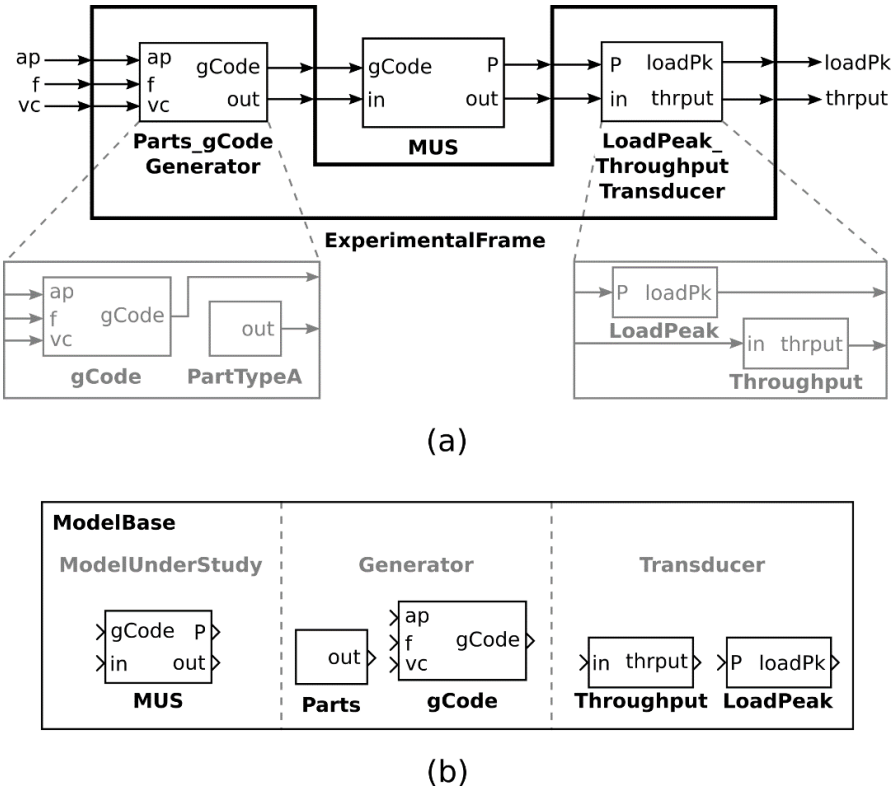


Abbildung 21: (a) Vereinfachtes Beispiel eines EF mit MUS. (b) Modellbibliothek mit den zugehörigen Komponenten des MUS und EF.

Wie in der Abbildung 21a dargestellt, können die Komponenten des EF hierarchisch aufgebaut werden. Ist zum Beispiel in weiteren Untersuchungen auch die aufgenommene elektrische Arbeit als Zielgröße von Interesse, so ist nur die Komponente *LoadPeak* im Transducer zu modifizieren. Weiterhin zeigt das Beispiel, das keinen *Acceptor* enthält, dass nicht immer alle drei EF-Komponenten zur Anwendung kommen müssen. Analog zum MUS oder dessen Komponenten können die Komponenten des EF ebenfalls in einer MB organisiert werden, wie Abbildung 21b zeigt. Im nächsten Unterabschnitt erfolgt eine detaillierte Diskussion zum Konzept des EF.

4.1.3 Zusammenfassung und Diskussion zum Experimental Frame

Ein wichtiger Aspekt der M&S ist die frühzeitige Trennung zwischen System und dem zugehörigen Kontext, welcher den Zusammenhang zwischen den Zielstellungen, Randbedingungen und weiteren Annahmen einer oder mehrerer zu untersuchender Problemstellungen repräsentiert. Das System und der Kontext können in der gleichen Art und Weise modular-hierarchisch modelliert werden und zu einem Simulationsmodell komponiert werden. Dabei wird das System als ein Model Under Study (MUS) und der Kontext als Experimental Frame (EF) im Simulationsmodell abgebildet. Beim Aufbau des EF werden drei grundlegende Komponenten: (i) Generator, (ii) Transducer und (iii) Acceptor unterschieden, wobei nicht in jedem Fall alle drei Komponenten zwingend enthalten sein müssen und die Kopplungsbeziehungen im EF variieren können.

Allgemein bildet der EF die Grundlage zur Durchführung eines Simulationslaufes. Zur Simulationslaufzeit interagiert das MUS mit dem EF. Der Generator generiert zeitbehaftete Einflussgrößen für das MUS, der Transducer sammelt und analysiert die zeitbehafteten

Ausgabegrößen des MUS und der Acceptor überprüft und bewertet die vom Transducer berechneten Zielgrößen an Hand einzuhaltender Randbedingungen und Toleranzen. Bei Nichteinhaltung der Randbedingungen kann der Acceptor den Simulationslauf vorzeitig beenden. Der Ansatz ermöglicht einen modular-hierarchischen Aufbau einfacher Simulationsexperimente, gemäß Unterabschnitt 2.2.2, in Form eines Simulationslaufes und bildet damit die Grundlage für alle in der Tabelle 2 aufgezeigten Experimentziele. Abbildung 22 zeigt eine Gegenüberstellung der im Unterabschnitt 2.2.2 beschriebenen Aufgaben zur Durchführung von Simulationsexperimenten und den Komponenten des EF.

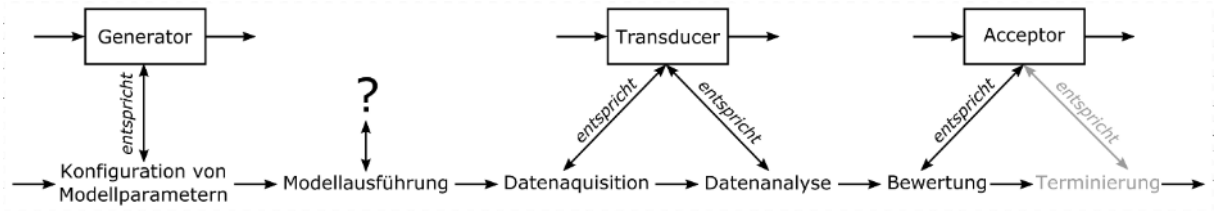


Abbildung 22: Zusammengefasste Gegenüberstellung der Aufgaben zur Durchführung von Simulationsexperimenten gemäß Abschnitt 2.2.2 und den Komponenten des EF.

Wie Abbildung 22 zeigt, können die meisten Aufgaben zur Durchführung von Simulationsexperimenten entsprechenden Komponenten des EF zugeordnet werden. Die einzige Ausnahme bildet die Aufgabe der Modellausführung, die die Auswahl eines Simulators, dessen Parametrierung und den Start eines Simulationslaufes umfasst. Das heißt, diese Aufgabe muss der Anwender manuell durchführen oder eine separate Softwarekomponente entwickeln. Trotzdem besitzt der Ansatz viel Potential in Bezug zum angestrebten Variantenmanagement von Simulationsexperimenten. Ein wesentlicher Aspekt ist die modular-hierarchische Struktur des EF, welche unmittelbar ein Variantenmanagement auf Basis des SES/MB Framework unterstützt. Die einzelnen Komponenten des EF können effizient ausgetauscht und somit an eine zu untersuchende Problemstellung angepasst werden. Demnach können auch mehrere MUS für ein EF und umgekehrt spezifiziert werden. Dieser Ansatz setzt natürlich eine klare Spezifikation der Schnittstellen der einzelnen Komponenten voraus. Weiterhin können die Komponenten des EF analog zu den Komponenten eines MUS in einer Modellbibliothek zur Wiederverwendung organisiert werden.

Aufbauend auf den bisher gewonnenen Erkenntnissen soll nachfolgend ein Variantenmanagement für komplexe Simulationsexperimente mit den übergeordneten Experimentzielen Screening, Sensitivitätsanalyse und Optimierung gemäß Abschnitt 2.2 entwickelt werden. Hierzu erfolgt im nächsten Abschnitt zunächst ein entsprechender Vorschlag zur modular-hierarchischen Strukturierung komplexer Simulationsexperimente.

4.2 Konzept zur Strukturierung und zum Ablauf komplexer Simulationsexperimente

Gemäß Unterabschnitt 2.2.2 bilden komplexe Simulationsexperimente einen Feedback-Prozess. Sie definieren sich durch eine Experimentmethode, die automatisiert Simulationsläufe durchführt, berechnete Simulationsergebnisse analysiert sowie speichert und gegebenenfalls weitere Simulationsläufe ausführt. Die Experimentmethode selbst ist ein software-

technisch realisiertes numerisches Verfahren, beispielsweise vom Typ Screening, Sensitivitätsanalyse oder Optimierung. Im Unterabschnitt 2.2.2 wurde die Strukturierung komplexer Simulationsexperimente stellvertretend am Beispiel der simulationsbasierten Optimierung diskutiert. Es stellte sich heraus, dass verschiedene Ansätze zur Strukturierung komplexer Simulationsexperimente in der Literatur existieren und diese teilweise konzeptionelle Unklarheiten aufweisen. Für das angestrebte Variantenmanagement komplexer Simulationsexperimente mit dem SES/MB Framework ist eine klare und wohldefinierte Strukturierung von essentieller Bedeutung. Wie im Kapitel 3 gezeigt, eignet sich das SES/MB Framework zum Variantenmanagement für modular-hierarchische Simulationsmodelle. Aus diesem Grund wird in diesem Abschnitt ein Konzept zur Strukturierung komplexer Simulationsexperimente unter Verwendung des modular-hierarchischen Ansatzes vorgestellt, wie in Abbildung 23 schematisch gezeigt.

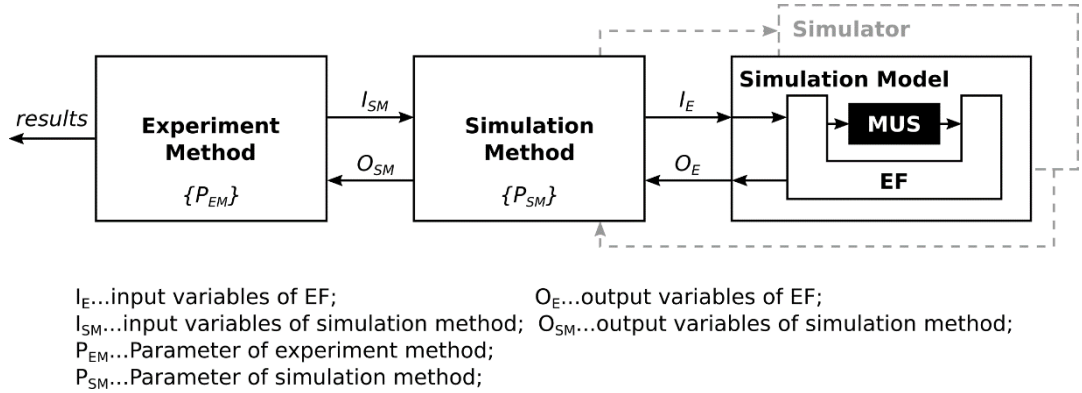


Abbildung 23: Konzeptionelles Blockschaltbild zur Strukturierung komplexer Simulationsexperimente.

Strukturell besteht ein komplexes Simulationsexperiment aus den Komponenten (i) *Experimentmethode*, (ii) *Simulationsmethode*, (iii) *Simulationsmodell* und (iv) *Simulator*. Der Simulator und die zugehörigen Einfluss- und Ausgabegrößen (gestrichelte Pfeile) sind nur der Vollständigkeit halber in die Abbildung 23 aufgenommen. Der Grund hierfür ist, dass der Simulator zwar auf der einen Seine eine essentielle Komponente bei der Durchführung von Simulationsexperimenten bildet, auf der anderen Seite aber in der Regel nur als parametrierbare numerische Methode vorliegt. Das ist speziell bei den kommerziellen M&S-Umgebungen, wie zum Beispiel MATLAB/Simulink, der Fall. Dem Anwender stehen nur die konfigurierbaren Parameter des Simulators und eine begrenzte Programmier-schnittstelle zur Verfügung. Gemäß Abbildung 23 wird zur Interaktion mit dem Simulator und dem Simulationsmodell die Komponente Simulationsmethode eingeführt. Sie hat die Funktion eines Wrappers. Sie setzt und liest beispielsweise simulatorspezifische Daten. Diese Interaktion mit dem Simulator wird im weiteren Verlauf implizit vorausgesetzt und nicht weiter betrachtet.

Da bereits im vorherigen Abschnitt der Aufbau des Simulationsmodells hinsichtlich der Trennung in Model Under Study (MUS) und Experimental Frame (EF) detailliert diskutiert wurde, wird im Weiteren nur auf die Komponenten Experimentmethode und Simulationsmethode aus Abbildung 23 näher eingegangen. Jede Komponente besitzt eine Schnittstelle mit Einflussgrößen und Ausgabegrößen, über die die Komponenten miteinander verknüpft sind. Die Bezeichner sind in Anlehnung an den EF mit $I_{\#}$ und $O_{\#}$ gewählt.

Weiterhin besitzen die Komponenten Experiment- und Simulationsmethode spezifische Parameter P_{EM} und P_{SM}. Diese werden einmalig vor jeder Experimentausführung gesetzt und bleiben über die gesamte Experimentlaufzeit konstant.

Die *Experimentmethode* entspricht einer numerischen Methode und besitzt keinen direkten Bezug zu einem Simulationsmodell oder Simulator, da diese in erster Linie für mathematische Problemstellungen in Form von Gleichungssystemen entwickelt wurden. Die Experimentmethode besitzt als Einflussgrößen die konfigurierbaren methodenspezifischen Parameter P_{EM} und die Ausgabegrößen O_{SM} der Simulationsmethode. Als Ausgabegrößen liefert sie die Eingabegrößen für die Simulationsmethode I_{SM} und die durch das Experiment zu berechnenden Zielgrößen *results*.

Die *Simulationsmethode* realisiert die Anbindung der Experimentmethode zum Simulationsmodell sowie dem Simulator und beschreibt wie Simulationsläufe durchzuführen sind. Demzufolge kann die Simulationsmethode nach Unterabschnitt 2.2.2 und Abbildung 4 als eine Experimentsteuerung betrachtet werden. Nach Abbildung 21 besitzt die Simulationsmethode I_{SM}, P_{SM} und O_E als Einflussgrößen und I_E sowie O_{SM} als Ausgabegrößen. Zur Laufzeit eines komplexen Simulationsexperimentes interagiert die Simulationsmethode mit der Experimentmethode und dem Simulator sowie dem Simulationsmodell. Somit muss die Simulationsmethode die konkrete Spezifikation zur Durchführung von Simulationsexperimenten codieren. Nachfolgend wird auf den allgemeinen Ablauf eines komplexen Simulationsexperimentes eingegangen.

4.2.1 Ablauf eines komplexen Simulationsexperimentes

Der schematische Ablauf eines komplexen Simulationsexperiments ist in Abbildung 24 unter Verwendung eines Sequenzdiagramms gezeigt.

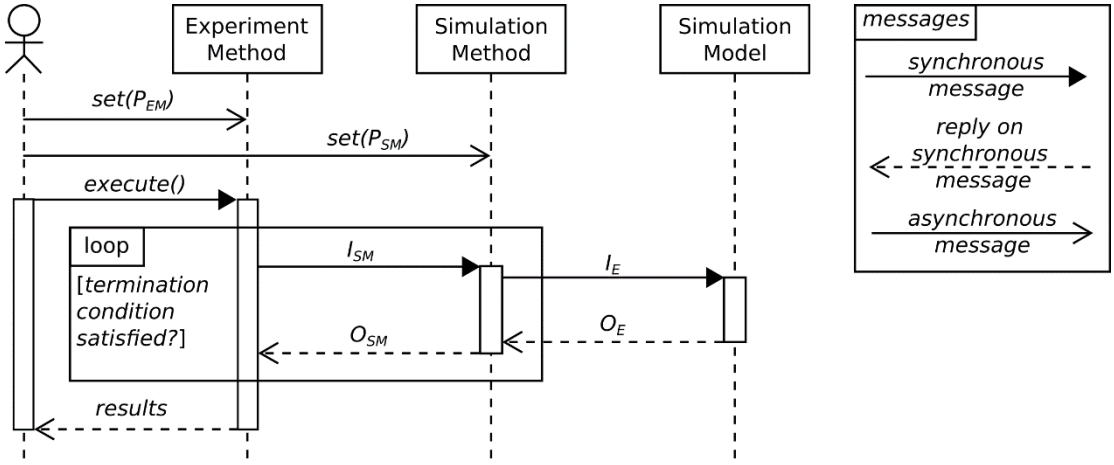


Abbildung 24: Sequenzdiagramm eines komplexen Simulationsexperimentes.

Am Anfang werden die methodenspezifischen Parametern P_{EM} und P_{SM} durch den Anwender gesetzt *(set(P_{EM})* und *set(P_{SM}))* und dann das Experiment ausgeführt *(execute())*. Die Experimentmethode generiert auf Basis der P_{EM} eine zulässige Konfiguration der Einflussgrößen I_{SM} und übermittelt diese an die Simulationsmethode. Diese parametriert auf Basis der P_{SM} den Simulator (nicht in der Abbildung 24 gezeigt) und startet die Abarbeitung des Simulationsmodells mit den Eingabegrößen I_E. Am Ende des Simulationslaufs gibt das Simulationsmodell die Ausgabegrößen O_E an die Simulationsmethode zurück, welche dann

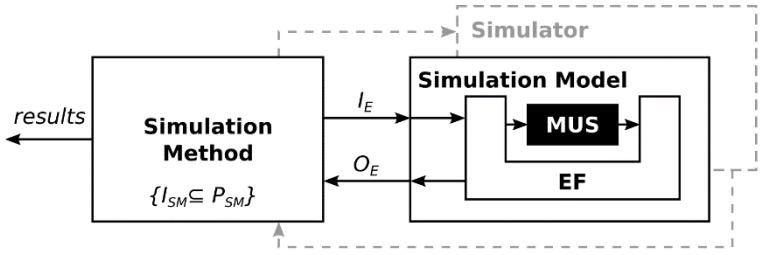
die O_{SM} berechnet und an die Experimentmethode zurückgibt. Letztere prüft, ob das für die Experimentmethode charakteristische Abbruchkriterium erreicht ist oder nicht. Ändert sich zum Beispiel im Rahmen einer Optimierung der Wert der Zielfunktion innerhalb eines vordefinierten Toleranzbereiches nicht mehr, so kann die Optimierung beendet werden. Bei einer Sensitivitätsanalyse kann die Anzahl der durchzuführenden Simulationsläufe als Abbruchkriterium definiert werden. Wird kein Abbruchkriterium erreicht, so generiert die Experimentmethode eine neue Konfiguration der Einflussgrößen I_{SM} und übermittelt diese erneut an die Simulationsmethode.

Ausgehend vom Aufbau und Ablauf komplexer Experimente können einfache Experimente als echte Untermenge komplexer Experimente betrachtet werden. Dieser Sachverhalt soll kurz gezeigt werden.

4.2.2 Einfache Simulationsexperimente als Untermenge komplexer Experimente

Der Aufbau eines komplexen Experimentes, gemäß Abbildung 23, kann durch einfache Modifikationen in ein *einfaches Experiment*, gemäß Abbildung 25, überführt werden. Die Experimentmethode entfällt und dadurch müssen die Schnittstellen der Simulationsmethode anders interpretiert werden. Die Ausgabegrößen (O_{SM}) der Simulationsmethode entsprechen unmittelbar den Zielgrößen *results*. Die von der Experimentmethode bereitgestellten Einflussgrößen (I_{SM}) können direkt mit den methodenspezifischen Parametern (P_{SM}) vereinigt werden, sodass die I_{SM} eine Teilmenge der P_{SM} bilden.

Ausgehend von der Abbildung 24 ändert sich der Ablauf eines *einfachen Experimentes* dahingehend, dass der Anwender vor einem Simulationslauf die P_{SM} konfiguriert und die Simulationsmethode ausführt. Diese parametriert auf Basis der P_{SM} den Simulator und bestimmt die I_E für das Simulationsmodell. Nach Beendigung des Simulationslaufs liefert das Simulationsmodell die Zielgrößen als O_E an die Simulationsmethode zurück. Diese bestimmt jetzt, ob noch weitere Simulationsläufe durchzuführen sind. Nach der Durchführung aller Simulationsläufe werden die Simulationsergebnisse an den Anwender zurückgeliefert.



I_E...input variables of EF; O_E...output variables of EF;
I_{SM}...input variables of simulation method; P_{SM}...Parameter of simulation method;

Abbildung 25: Ableitung eines einfachen Experiments aus einem komplexen Experiment nach Abbildung 23.

In Anlehnung an den Unterabschnitt 2.2.2 können einfache Experimente zum Beispiel für die Durchführung von explorativen Analysen herangezogen werden. Da einfache Experimente eine echte Untermenge komplexer Experimente bilden, wird nachfolgend nur ein komplexes Experiment am Beispiel einer Optimierung vorgestellt.

4.2.3 Beispiel eines komplexen Simulationsexperimentes

Abbildung 26 zeigt schematisch die Anwendung des vorgestellten Konzepts zur Strukturierung komplexer Simulationsexperimente am Beispiel einer Optimierung. Der in Abbildung 23 gezeigte Simulator sowie die Kopplung der Simulationsmethode mit dem Simulator ist nicht explizit dargestellt, wird aber implizit mitbetrachtet.

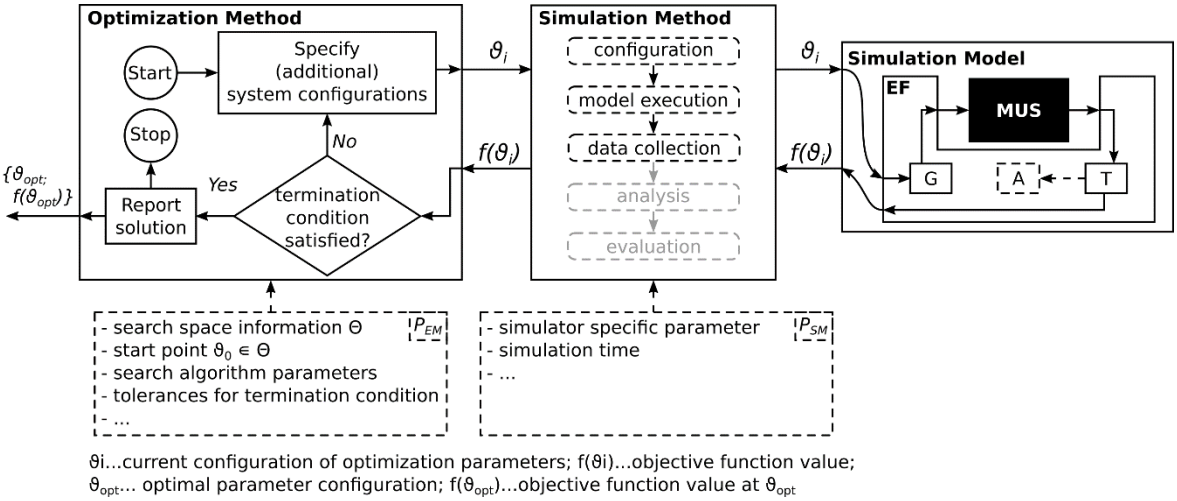


Abbildung 26: Strukturierung komplexer Simulationsexperimente am Beispiel der Optimierung nach dem Konzept in Abbildung 23.

Die *Experimentmethode* entspricht einer konkreten numerischen Optimierungsmethode. Vor der Experimentausführung wird diese mit den methodenspezifischen Parametern P_{EM} initialisiert, wie in Abbildung 26 schematisch angedeutet. Unter anderem zählen dazu Informationen zum Suchraum Θ und ein Startpunkt $\vartheta_0 \in \Theta$. Die Optimierungsmethode erhält von der Simulationsmethode nach jedem Simulationslauf als O_{SM} den aktuellen Zielfunktionswert $f(\vartheta_i)$ und stellt die neu berechnete Parameterkonfiguration $\vartheta_i \in \Theta$ als I_{SM} der Simulationsmethode bereit. Die Zielgrößen *results* am Ende des Experiments bildet das Tupel $\{\vartheta_{opt}, f(\vartheta_{opt})\}$ mit der optimalen Parameterkonfiguration ϑ_{opt} und den dazu gehörigen Zielfunktionswert $f(\vartheta_{opt})$.

Die *Simulationmethode* realisiert die Anbindung der Optimierungsmethode an das Simulationsmodell inklusive Simulator und vermittelt während der Experimentdurchführung zwischen den Komponenten. Analog zur Experimentmethode werden gemäß Abbildung 26 vor der Experimentausführung die methodenspezifischen Parameter P_{SM}, zu denen auch die Einstellgrößen des Simulators zählen, initialisiert. Wie bereits in den vorherigen Abschnitten diskutiert, beschreibt die Simulationsmethode die automatisierte Durchführung eines oder mehrerer Simulationsläufe. In Analogie zum Unterabschnitt 2.2.2 wird in Abbildung 26 auf die dort eingeführten Aufgaben zurückgegriffen werden. Die Aufgabe *Konfiguration* überführt die Eingangsgrößen I_{SM} der Simulationsmethode, die aktuellen ϑ_i, in Eingangsgrößen I_E für das Simulationsmodell. In diesem Fall können die I_{SM} ohne weitere Modifikationen einfach durchgeleitet werden. Als nächstes erfolgt die *Modellausführung*, also auf Basis der methodenspezifischen Parameter P_{SM} wird der Simulator parametriert und ein Simulationslauf gestartet. Die Aufgabe *Datenakquisition* sammelt die O_E vom Simulationsmodell und überführt diese als O_{SM} in das von der Optimierungsmethode

benötigte Format. Wie in Abbildung 26 gezeigt, können in dem Fall auch die Ausgabegrößen O_E, besser gesagt der aktuelle Zielfunktionswert $f(\vartheta_i)$, ohne Modifikation als O_{SM} an die Experimentmethode durchgeleitet werden.

Die Aufgaben *Analyse* und *Bewertung* sind vollständigkeitshalber aufgenommen, für dieses Beispiel aber nicht grundlegend notwendig, da die Durchführung dieser Aufgaben bereits durch die Komponenten Transducer und Acceptor des Experimental Frame im Simulationsmodell übernommen werden. So erfolgt beispielsweise die Berechnung des aktuellen Zielfunktionswertes $f(\vartheta_i)$ im Transducer. Jedoch könnten diese Aufgaben in der Simulationsmethode im Sinne des Vier-Augen-Prinzips zur Durchführung von Mehrfachkontrollen einzelner Ergebnisse von Simulationsläufen verwendet werden.

Der Ablauf der Optimierung erfolgt analog zum allgemeinen Ablauf eines komplexen Simulationsexperimentes in Abbildung 24. Die bis hier eingeführte Struktur komplexer Simulationsexperimente bildet die Grundlage für das nachfolgend diskutierte Variantenmanagement von Simulationsexperimenten unter Verwendung des SES/MB Frameworks.

4.3 Variantenmanagement von Simulationsexperimenten mit dem SES/MB Framework

In diesem Abschnitt wird das Variantenmanagement von Simulationsexperimenten auf Basis der eingeführten Struktur komplexer Experimente, nachfolgend nur als Experiment bezeichnet, und dem System Entity Structure/Model Base (SES/MB) Framework vorgestellt. Abbildung 27 zeigt das Vorgehensmodell, aufbauend auf dem Vorgehensmodell zum SES/MB Framework in Abbildung 12, unter Beachtung der wesentlichen Phasen des Variantenmanagements nach Abbildung 18. Auf den ersten Blick zeigt das modifizierte Vorgehensmodell nur marginale Abweichungen zur Darstellung in Abbildung 12. Jedoch berücksichtigt das Vorgehensmodell und das SES/MB Framework im Abschnitt 3.1 nur die Variantenanalyse, Variantenimplementierung und Variantengenerierung von Simulationsmodellen im Sinne unterschiedlicher *Model Under Study* (MUS) Varianten. Die Abbildung 27 erweitert das Variantenmanagement auf komplexe Experimente gemäß Abbildung 23. Das heißt, bei der Variantenanalyse und Variantenformalisierung sind nicht nur die Komponenten und modular-hierarchischen Strukturen des MUS zu berücksichtigen, sondern analog dazu auch die der Experimentmethode, der Simulationsmethode und des Experimental Frame. Ebenso müssen in der Phase der Variantenimplementierung neben den Strukturen der MUS-Varianten auch die Varianten der Experimentstrukturen in einer SES abgebildet werden. Gleiches gilt für die Implementierung und Organisation aller benötigten elementaren Komponenten, wie in der MB in Abbildung 27 prinzipiell gezeigt. Natürlich muss auch die formale Kopplung zwischen der SES und der MB auf alle Elemente erweitert werden. Die beiden Methoden *pruning* und *translation* sind ebenfalls entsprechend anzupassen, so dass diese die Ableitung einer konkreten Experimentstruktur in Form einer Pruned Entity Structure (PES) und das Generieren einer ausführbaren Experimentvariante auf Basis der PES und der MB unterstützen.

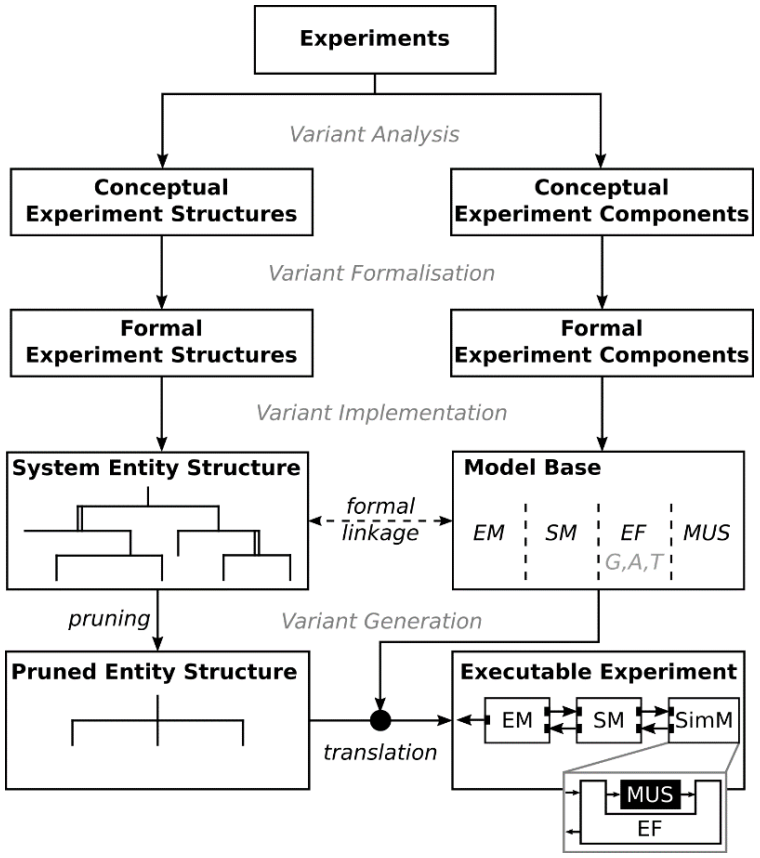


Abbildung 27: Vorgehensmodell für ein Variantenmanagement komplexer Experimente auf Basis des SES/MB Frameworks.

Die in Abbildung 27 gezeigte Vorgehensweise kann natürlich hinterfragt werden. Denn den Ausgangspunkt eines M&S-Projektes, auch im Rahmen der Variantenvielfalt, bildet gemäß Abbildung 1 und 18 die *Zielbeschreibung*. Demnach sind die durchzuführenden Experimentvarianten nach Abbildung 23 erst einmal unbekannt. Das Vorgehensmodell in Abbildung 27 setzt diese aber für das Variantenmanagement komplexer Experimente voraus. Somit müssen in der Phase *Variantenanalyse* zunächst sukzessiv die durchzuführenden Experimentvarianten konzeptionell realisiert werden, um dann im Anschluss die tatsächlich notwendigen Experimentstrukturen mit den zugehörigen Komponenten zu identifizieren. Für die sukzessive Variantenanalyse wird in der Abbildung 28 ein Vorschlag aufgezeigt.

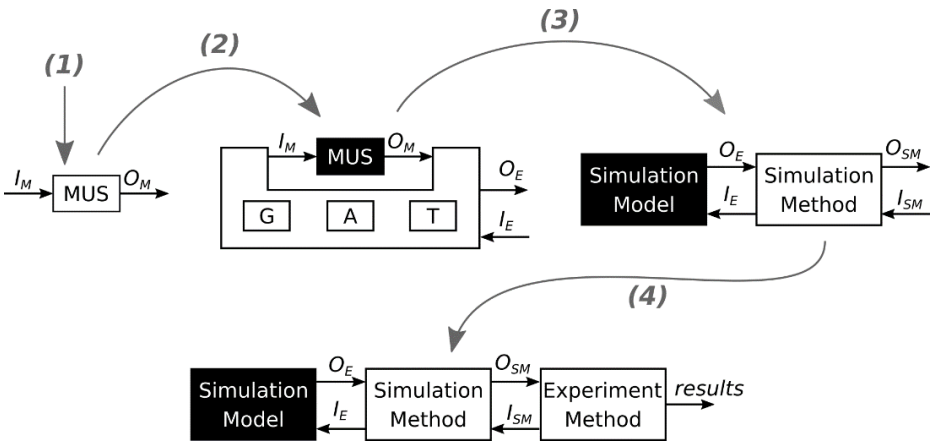


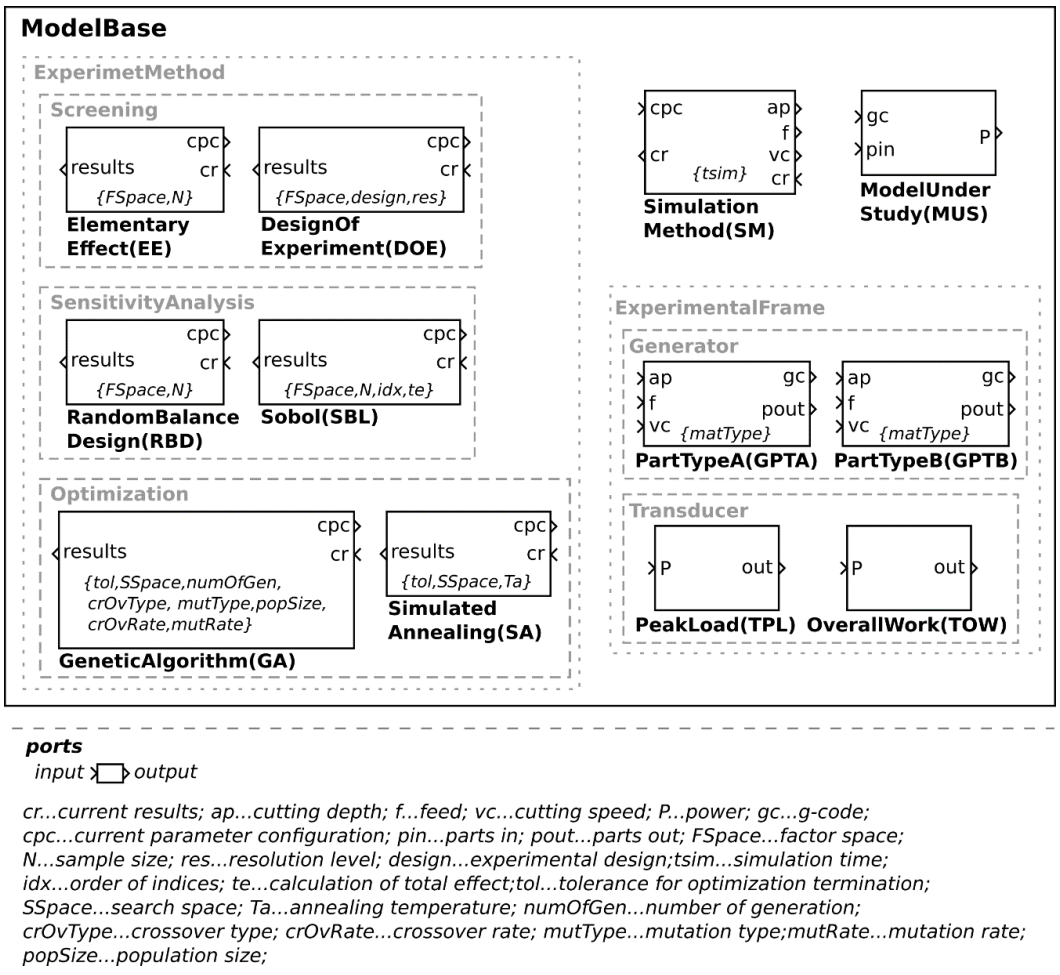
Abbildung 28: Vorschlag zur sukzessiven Variantenanalyse komplexer Experimente.

Demnach erfolgt zunächst eine Variantenanalyse des MUS mit den zugehörigen Schnittstellen I_M und O_M. Im zweiten Schritt der Variantenanalyse werden entsprechend der Problemstellung die erforderlichen EF mit den I_E und O_E abgeleitet. Danach werden die notwendigen Simulationsmethoden zum Durchführen von Simu lationsläufen, besser gesagt die P_{SM}, I_{SM} und O_{SM} identifiziert und abschließend werden die ausgewählten Experimentmethoden konzeptionell abgeleitet.

In den folgenden drei Unterkapiteln werden wesentliche Aspekte des Variantenmanagements komplexer Experimente näher betrachtet. Zuerst wird beispielhaft auf die Organisation experimentspezifischer Komponenten in einer MB eingegangen. Anschließend wird mit Bezug auf die eingeführte MB die Spezifikation von Experimentstrukturen mit einer SES erläutert und danach die Generierung ausführbarer Experimente diskutiert. Im späteren Anwendungskapitel wird die gesamte Problematik nochmals aufgegriffen.

4.3.1 Organisation der experimentspezifischen Komponenten in einer Modellbibliothek

Abbildung 29 zeigt beispielhaft eine Modellbibliothek mit konfigurierbaren Komponenten zur Untersuchung von fertigungstechnischen Prozesskettenmodellen.



cr...current results; ap...cutting depth; f...feed; vc...cutting speed; P...power; gc...g-code; cpc...current parameter configuration; pin...parts in; pout...parts out; FSpace...factor space; N...sample size; res...resolution level; design...experimental design;tsim...simulation time; idx...order of indices; te...calculation of total effect;tol...tolerance for optimization termination; SSpace...search space; Ta...annealing temperature; numOfGen...number of generation; crOvType...crossover type; crOvRate...crossover rate; mutType...mutation type;mutRate...mutation rate; popSize...population size;

Abbildung 29: Beispielhafte und vereinfachte Darstellung einer Modellbibliothek zur Generierung komplexer Experimente für fertigungstechnische Prozesskettenmodelle.

Die Komponenten besitzen wohldefinierte Ein- und Ausgabeschnittstellen zur Komposition komplexer Experimente für die drei Experimentziele Screening, Sensitivitätsanalyse und Optimierung. Zur Vereinfachung wurden nur ein MUS und nur eine Komponente vom Typ Simulationsmethode ausgewählt. Das MUS entspricht einem konkreten Prozesskettenmodell gemäß Abbildung 15 in Unterabschnitt 3.2.2. Darüber hinaus wurde zur Vereinfachung keine Acceptor Komponente in die MB aufgenommen.

Zur Untersuchung der drei Experimentziele wurden für jede Experimentmethode stellvertretend jeweils zwei numerische Verfahren mit entsprechenden methoden-spezifischen Parametern ausgewählt, welche in Tabelle 7 zusammengefasst sind. Für das Screening wurden die Methoden *Elementary Effect* (EE) und *Design of Experiment* (DOE) gemäß Saltelli et al. [132] ausgewählt. Zur Sensitivitätsanalyse wurden die Methoden *Random Balance Design* (RBD) und *Sobol* (SLB) nach [132] gewählt und für die Optimierung wurden stellvertretend zwei Metaheuristiken, der *Genetische Algorithmus* (GA) und das *Simulated Annealing* (SA*)* nach Sumathi, S., Surekha [155], gewählt. In Abbildung 29 sind die methodenspezifischen Parametermengen P_{EM} und P_{SM} (vgl. Abbildung 23) jeweils in geschweiften Klammern schematisch angegeben. Im Weiteren erfolgt keine detaillierte Erläuterung der methodenspezifischen Parameter. Hierzu sei auf die Literatur [132, 155] verwiesen.

Zur Komposition des Experimental Frame, welcher den Kontext des MUS codiert, wurden zwei unterschiedliche Generatoren und Transducer in die MB aufgenommen. Erstere unterscheiden sich durch die Generierung unterschiedlicher Bauteile. Die Transducer ermöglichen die Analyse der Ausgabegrößen des MUS hinsichtlich auftretenderLastspitzen (*PeakLoad*) und der insgesamt aufgenommenen elektrischen Arbeit (*OverallWork*). Die beispielhafte Spezifikation von Experimentstrukturen mit einer SES wird im nächsten Unterabschnitt thematisiert.

Tabelle 7: Übersicht zu den ausgewählten Experimentmethoden und deren Parametern.

<table>
<tr><th colspan="4">Screening nach [132]</th></tr>
<tr><td>Experimentmethode</td><td colspan="2">Elementary Effect</td><td>Design of Experiment</td></tr>
<tr><td rowspan="3">Methodenspezifische Parameter (P_{EM})</td><td colspan="3">factor space (FSpace) = {...}</td></tr>
<tr><td rowspan="2">sample size (N) = 500</td><td>1)</td><td>design = full</td></tr>
<tr><td>2)</td><td>design = fractional
resolution (res) = 4</td></tr>
<tr><th colspan="4">Sensitivitätsanalyse nach [132]</th></tr>
<tr><td>Experimentmethode</td><td colspan="2">Random Balance Design</td><td>Sobol</td></tr>
<tr><td rowspan="2">Methodenspezifische Parameter (P_{EM})</td><td colspan="3">factor space (FSpace) = {...}</td></tr>
<tr><td colspan="2">sample size (N) = 500</td><td>sample size (N) = 500
indices (idx) = first order
total effect (te) = yes</td></tr>
</table>

<table>
<tr><th colspan="4">Optimierung nach [155]</th></tr>
<tr><td>Experimentmethode</td><td colspan="2">Genetic Algorithm</td><td>Simulated Annlealing</td></tr>
<tr><td rowspan="4">Methodenspezifische Parameter (P_{EM})</td><td colspan="3">search space (SSpace) = {...}, tolerance (tol) = 1e-3</td></tr>
<tr><td colspan="2">number of generation (numOfGen) = 1000
cross over type (crOvType) = two point
mutation Type (mutType) = bit flip</td><td rowspan="3">initial temperature (Ta) = 1000</td></tr>
<tr><td>1)</td><td>population size (popSize) = 50
cross over rate (crOvRate) =0.6
mutation rate (mutRate) = 1e-3</td></tr>
<tr><td>2)</td><td>population size (popSize) = 30
cross over rate (crOvRate) =0.9
mutation rate (mutRate) = 1e-2</td></tr>
</table>

4.3.2 Spezifikation unterschiedlicher Experimentstrukturen in einer SES

Mit Bezug auf die beispielhafte Modellbibliothek sind in der SES in Abbildung 30 insgesamt 32 Varianten möglicher Experimentstrukturen zur Untersuchung der drei angestrebten Experimentziele: (i) Screening, (ii) Sensitivitätsanalyse und (iii) Optimierung spezifiziert. Die Struktur jedes einzelnen Experiments orientiert sich am grundlegenden Aufbau komplexer Experimente gemäß Abbildung 23. Die Syntax und Semantik der SES entspricht den Ausführungen im Kapitel 3, einschließlich der dort eingeführten Erweiterungen.

Bis auf die Wurzel-Entität besitzen alle weiteren Knoten konkrete Attribute, die in geschweiften Klammern angegeben sind. Die formale Kopplung zwischen der Modellbibliothek und den einzelnen Entitäten der SES ist im charakteristischen Attribut *mb* definiert. Es sei darauf hingewiesen, dass dieses nicht in jedem Fall an den Blatt-Entitäten definiert ist, da es teilweise auf Grund von Spezialisierungen im aktuellen Pfad vererbt wird. Die weiteren Attribute der einzelnen Entitäten-Knoten repräsentieren Wertzuweisungen an Parameter in Übereinstimmung mit den Konfigurationsmöglichkeiten der Komponenten in der Modellbibliothek.

Gemäß der grundlegenden Struktur komplexer Experimente nach Abbildung 23 wird die Wurzel-Entität *ComplexExperiment* in die Entitäten *ExperimentMethod*, *SimulationMethod* und *SimulationModel* zerlegt. Die Entität *SimulationModel* wird weiter in *ModelUnderStudy* und *ExperimentalFrame* dekomponiert. Wobei letztere in *Generator* und *Transducer* zerlegt wird. Die Aspekt-Knoten *Experiment-Dec*, *SimulationModel-Dec* und *ExperimentalFrame-Dec* spezifizieren die notwendigen Kopplungsrelationen zwischen den jeweiligen Entitäten. Die für das Variantenmanagement essentiellen Variationspunkte werden in der Abbildung 30 ausschließlich durch deskriptive Knoten vom Typ Spezialisierung ausgedrückt und sind in Tabelle 8 noch einmal zusammengefasst.

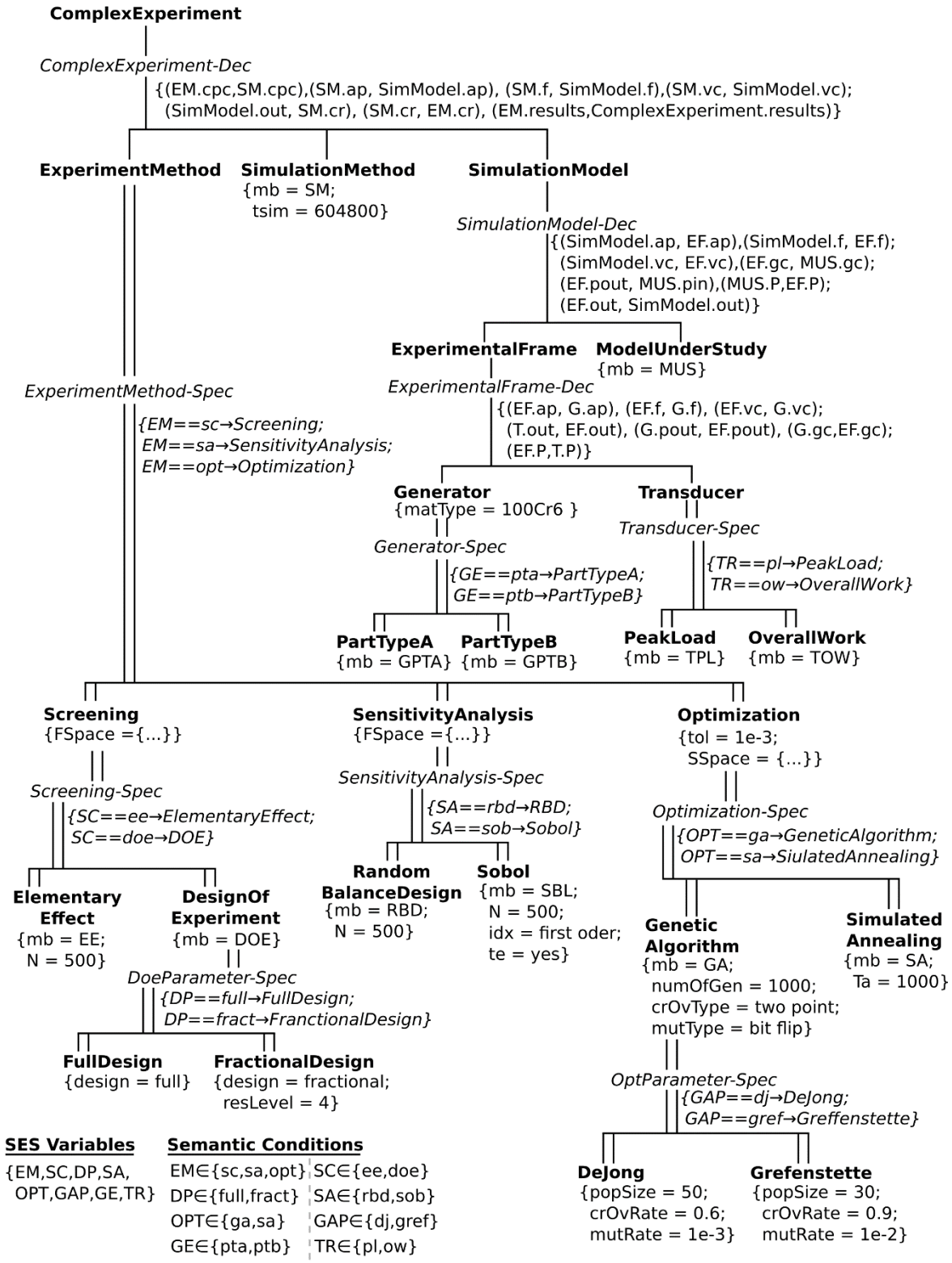


Abbildung 30: Spezifikation von 32 Strukturen komplexer Experimente mit einer SES gemäß der grundlegenden Struktur komplexer Experimente in Abbildung 23.

Tabelle 8: Zusammenfassung der Variationspunkte der SES in Abbildung 30.

Variationspunkte	**Auswahlmöglichkeiten**	
	qualitativ	*quantitativ*
ExperimentMethod-Spec	Screening, SensitivityAnalysis, Optimization	3
Generator-Spec	PartTypeA, PartTypeB	2
Transducer-Spec	PeakLoad, OverallWork	2
Screening-Spec	ElementaryEffect, DesignOfExperiment	2
DoeParameter-Spec	FullDesign, FractionalDesign	2
SensitivityAnalysis-Spec	RandomBalanceDesign, Sobol	2
Optimization-Spec	GeneticAlgorithm, SimulatedAnnealing	2
OptParameter-Spec	DeJong, Grefenstette	2

Die Variationspunkte *DoeParameter-Spec* und *OptParameter-Spec* stellen hierbei eine Besonderheit dar. Der jeweilige Variationspunkt bezieht sich ausschließlich auf die Parametrierung der zugehörigen Entität. Beispielsweise werden bei der Optimierung mit dem Genetischen Algorithmus die gemeinsamen Parameter in der Entität *GeneticAlgorithm* aufgelistet und Unterschiede der Parametrierung durch die Kinder *DeJong* und *Grefenstette* der Spezialisierung *OptParameter-Spec* ausgedrückt. Auf diese Art und Weise können zum Beispiel unterschiedliche Parameterkonfigurationen der Optimierungsmethode und deren Auswirkung auf das Ergebnis einer Optimierung untersucht werden. Die Generierung ausführbarer Experimentvarianten wird im nächsten Unterabschnitt beschrieben.

4.3.3 Generierung einer ausführbarer Experimentvariante

Für die Generierung einer ausführbaren Experimentvariante muss zunächst die Variabilität einer SES aufgelöst werden. Dies erfolgt im ersten Schritt durch Wertzuweisungen an die SES Variablen. Die SES Variablen bilden eine Nutzerschnittstelle und sind in der Abbildung 30 unten links definiert. Ihre zulässigen Wertebereiche sind in Form von *Semantic Conditions* angegeben. Unter Verwendung der Methode *pruning* werden alle Variationspunkte aufgelöst, wenn zuvor die Semantic Conditions zu true evaluiert wurden. Das Ergebnis ist eine Pruned Entity Structure (PES). Für die SES Variablen Belegung:

- *EM = sc, SC = ee, GE = ptb, TR = pl*

ist die resultierende PES in Abbildung 31 gezeigt. An dieser Stelle sei noch einmal auf das Axiom Vererbung, AX6 in Unterabschnitt 3.2.2, hingewiesen, welches im Zusammenhang mit dem deskriptiven Spezialisierungs-Knoten und dessen Vater-Entität sowie Kind-Entitäten steht. Aufgrund dieses Axioms erfolgt beim *Pruning* eine Anpassung der Entitäten-

Namen und der Kopplungsrelationen. Weiterhin werden die Attribute vererbt, wie es zum Beispiel an den Entitäten *PartTypeB_Generator* oder *ElementaryEffect_Screening_ExperimentMethod* zu sehen ist.

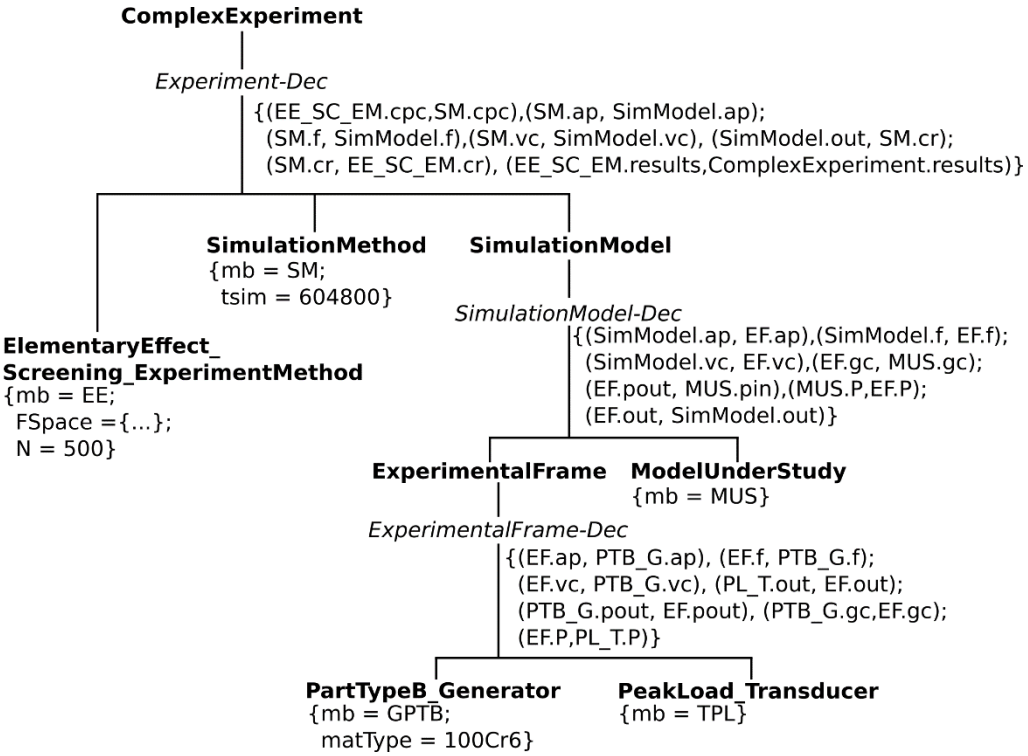


Abbildung 31: Eine mögliche PES der SES in Abbildung 30.

Nach erfolgreichem Pruning kann mit der Methode *translation*, unter Verwendung der PES und der Modellbibliothek eine ausführbare Variante eines komplexen Experiments generiert werden. Abbildung 32 zeigt schematisch das generierte Simulationsexperiment basierend auf der PES in Abbildung 31 und der MB in Abbildung 29. Das komplexe Experiment ist vollständig konfiguriert und kann durch den Anwender manuell oder mittels einer separaten Softwarekomponente ohne weitere Modifikationen ausgeführt werden.

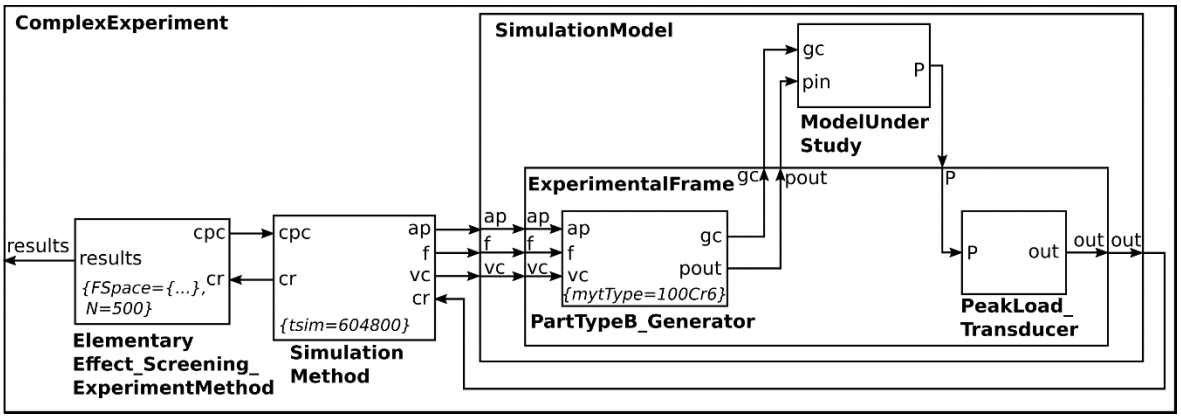


Abbildung 32: Durch die Methode *translation* generierte Variante eines ausführbaren komplexen Experiments.

Aufgrund des im vorherigen Abschnitt eingeführten Konzepts zur Strukturierung komplexer Experimente kann das SES/MB Framework ohne weitere Modifikationen unmittelbar zum

Variantenmanagement von Experimenten eingesetzt werden. Im Anschluss erfolgt eine Zusammenfassung wesentlicher Aspekte dieses Kapitels.

4.4 Zusammenfassung

Wie bereits im Kapitel 3 postuliert, ist eine klare Strukturierung eine essentielle Voraussetzung für das Variantenmanagement mit dem SES/MB Framework. Am Anfang dieses Kapitels wurde zunächst eine alternative Strukturierung eines Simulationsmodells vorgeschlagen. Dabei wurde auf bewährte Konzepte der Systemtheorie zurückgegriffen. Der Ansatz basiert auf der frühzeitigen Trennung des zu untersuchenden Systems und dem Kontext, welcher im Wesentlichen die Zielstellung und Randbedingungen der zu bearbeitenden Problemstellung repräsentiert. Hierbei wird das System als *Model Under Study* (MUS) und der zugehörige Kontext als *Experimental Frame* (EF) modelliert. Beide werden durch zwei disjunkte miteinander gekoppelte Komponenten im Simulationsmodell abgebildet. Diese Trennung kann zur Steigerung der Flexibilität und Effektivität bei der Durchführung von Simulationsläufen führen. Weiterhin kann sowohl das MUS als auch der EF für weitere Projekte wiederverwendet werden. Bezogen auf den Abschnitt 2.2 bildet der EF die Basis für einfache Simulationsexperimente und damit die Grundlage für alle in der Tabelle 2 gezeigten Experimentziele.

Im nächsten Schritt erfolgte ein neuer Vorschlag zur Strukturierung komplexer Experimente. Dabei wird ein komplexes Experiment in drei miteinander gekoppelte Komponenten zerlegt: (i) Experimentmethode, (ii) Simulationsmethode und (iii) Simulationsmodell. Die Experimentmethode ist eine softwaretechnisch umgesetzte numerische Methode, beispielsweise vom Typ Screening, Sensitivitätsanalyse oder Optimierung. Die Simulationsmethode bildet die Schnittstelle zwischen dem Simulationsmodell, der Experimentmethode und dem Simulator. Zur Experimentlaufzeit interagiert die Simulationsmethode mit der Experimentmethode und mit dem Simulationsmodell. Die Simulationsmethode konfiguriert das Simulationsmodell und den Simulator, führt einen Simulationslauf durch, erhält Simulationsergebnisse vom Simulationsmodell und übermittelt diese an die Experimentmethode.

Auf Basis der neu vorgeschlagenen Struktur wurde das Variantenmanagement komplexer Experimente unter Verwendung des SES/MB Frameworks an einem Beispiel vorgestellt. Dabei konnte gezeigt werden, dass das SES/MB Framework ohne zusätzliche Modifikationen zum Variantenmanagement komplexer Experimente angewendet werden kann.

Mit dem aufgezeigten Ansatz können ausführbare komplexe Experimente generiert werden. Bisher entscheidet jedoch der Anwender interaktiv, welche Experimente in welcher Reihenfolge zu generierten und auszuführen sind. Am Rande der bisherigen Ausführungen wurde bereits mehrmals erwähnt, dass auch eine Kopplung mit einer separaten Softwarekomponente erfolgen kann. Dazu wird im nächsten Kapitel ein *Experimentation Framework* vorgestellt, welches eine weitgehende Automatisierung des Experimentierprozesses ermöglicht.

5 Automatisiertes Experimentieren auf Basis des Variantenmanagements

Im vorhergehenden Kapitel wurden die Grundlagen für das Variantenmanagement von Experimenten unter Verwendung des SES/MB Frameworks geschaffen. Die MB organisiert dabei die für das Erstellen eines komplexen Experimentes notwendigen Komponenten, einschließlich Experimentmethoden, und die SES spezifiziert die Menge der zulässigen Experimentstrukturen. Die betrachteten komplexen Experimente beziehen sich auf unterschiedliche Experimentphasen, welche in Tabelle 2 im Abschnitt 2.2 in kurzfristig, mittelfristig und langfristig klassifiziert wurden. Zur Generierung einer ausführbaren Variante eines komplexen Experimentes müssen konkrete Wertzuweisungen an SES-Variablen, die Benutzerschnittstelle der SES, erfolgen. Bisher wurde dieser Aspekt als manueller Schritt durch einen Anwender betrachtet.

In diesem Kapitel wird ein Ansatz zum automatisierten Experimentieren erarbeitet. Zuerst wird ein Konzeptrahmen vorgestellt. Dieser ist in allen drei Experimentphasen, nachfolgend teilweise vereinfacht nur als Phasen bezeichnet, anwendbar. Hinsichtlich der Anwendbarkeit auf die drei eingeführten Klassen zum Experimentaufbau wird im ersten Abschnitt nur Bezug auf komplexe Experimente und die Unterklasse der einfachen Experimente genommen. Im anschließenden Kapitel wird die Übertragung auf hochkomplexe Experimente am Beispiel einer kombinierten Parameter- und Strukturoptimierung vorgestellt. Abschließend erfolgt eine kurze Kapitelzusammenfassung.

5.1 Konzeptrahmen zum automatisierten Experimentieren

Die Grundidee für das automatisierte Experimentieren mit Experimentvarianten ist den einfachen Experimenten aus dem Unterabschnitt 2.2.2 entlehnt. Demnach besteht ein *einfaches Experiment* aus einer (i) *Experimentsteuerung,* (ii) einem *Simulationsmodell* und (iii) einem *Simulator*. Die Experimentsteuerung entspricht einer ausführbaren Experimentspezifikation und beschreibt detailliert das am Simulationsmodell durchzuführende Experiment unter Verwendung eines Simulators. Dabei wird das Simulationsmodell anhand einer Parameterkonfiguration Pc (vgl. Gleichung 2) aus dem zugehörigen Parameterraum P (vgl. Gleichung 1) konfiguriert und ein Simulationslauf gestartet. Nach dem Simulationslauf werden die Simulationsergebnisse analysiert, bewertet und dem Anwender präsentiert. Dieses Vorgehen wird jetzt auf ein höheres Abstraktionsniveau übertragen, um das automatisierte Experimentieren mit komplexen Experimentvarianten phasenübergreifend zu ermöglichen.

Die SES-Variablen des SES/MB Frameworks bilden gemäß Kapitel 3 die Benutzerschnittstelle. Damit eine ausführbare Experimentvariante generiert werden kann, müssen zulässige Wertzuweisungen an die SES-Variablen erfolgen und diese an die *pruning* Methode des SES/MB Frameworks übergeben werden. Demnach muss eine konkrete SES-Variablenbelegung genau eine konkrete Experimentvariante kodieren. In diesem Kontext können die SES-Variablen als Parameter der SES angesehen werden. In Analogie zur Betrachtung des Parameterraumes bei einfachen Experimenten in Unterabschnitt 2.1.2 (vgl. Gleichung 1) bildet das Kreuzprodukt der Wertebereiche der SES-Variablen den Parameterraum der SES. Dieser Sachverhalt ist in Gleichung 13 gezeigt, wobei $P_{SESVar,i}$ für den Wertebereich der i-ten SES-Variablen steht.

$$P_{SESVar} = P_{SESVar,1} \times P_{SESVar,2} \times \ldots \times P_{SESVar,n} \quad (13)$$

Demgemäß entspricht nach Gleichung 2 in Unterabschnitt 2.1.2 eine konkrete Wertebelegung der SES-Variablen, wie in Gleichung 14, einer konkreten Parameterkonfiguration P_c. Analog können Mengen von Parameterkonfigurationen, wie in Gleichung 15 dargestellt, definiert werden.

$$P_{SESVar,c} = \left(p_{SESVar,1}, p_{SESVar,2}, \ldots, p_{SESVar,n}\right) \quad (14)$$

$$\left\{P_{SESVar,c}\right\} = \{P_{SESVar,c,1}, P_{SESVar,c,2}, \ldots, P_{SESVar,c,n}\} \quad (15)$$

Jede Parameterkonfiguration $P_{SESvar,c,i}$ codiert eine zu generierende Experimentvariante. Den zentralen Kern des automatisierten Experimentierens mit Experimentvarianten bildet *jetzt* die Untersuchung des Parameterraumes der SES-Variablen. Der Konzeptrahmen für das angestrebte automatisierte Experimentieren ist in Abbildung 33 aufgezeigt.

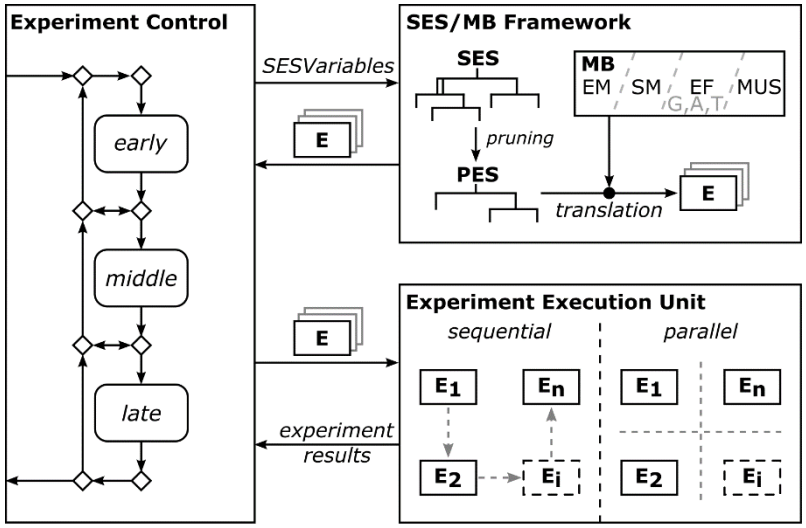


Abbildung 33: Konzeptrahmen für das automatisierte Experimentieren.

Der Konzeptrahmen basiert auf den drei Hauptkomponenten: (i) *Experimentsteuerung,* (ii) *SES/MB Framework* und (iii) *Experimentausführungseinheit*. Das SES/MB Framework beschreibt alle zulässigen Experimentvarianten gemäß den Ausführungen in Abschnitt 4.3. Die Generierung der gewünschten Experimentvarianten erfolgt anhand der SES-Variablenbelegungen, getreu Gleichung 14 sowie 15, durch die Methoden *pruning* und *translation* des SES/MB Frameworks. Die Experimentsteuerung und Experimentausführungseinheit werden nachfolgenden näher beschrieben.

5.1.1 Experimentsteuerung für die automatisierte Experimentausführung

Die Experimentsteuerung codiert eine detaillierte Beschreibung zur automatisierten Experimentdurchführung und interagiert während des Experimentprozesses mit dem SES/MB Framework sowie mit der Experimentausführungseinheit.

Aus übergeordneter Sicht umfasst die Experimentsteuerung:

- die Verwaltung des SES-Variablenraumes nach Gleichung 13,
- die Verwaltung der vom SES/MB Framework generierten Experimente,
- die Spezifikation einer phasenbezogenen Experimentdurchführung und
- die Spezifikation einer Ablaufsteuerung für alle durchzuführenden Experimente.

Getreu Abbildung 33 kann die Experimentsteuerung eine phasenbezogene Experimentdurchführung unterstützen. In diesem Fall ist für jede Phase: (i) *frühzeitig*, (ii) *mittelfristig* und (iii) *langfristig* jeweils eine eindeutige Teilsteuerung zu erstellen. Eine übergeordnete Ablaufsteuerung sorgt für die Durchführung der Experimente in einer logischen Reihenfolge.

Zur Spezifikation der gesamten Experimentsteuerung oder einer phasenbezogenen Steuerung können die in Unterabschnitt 2.2.2 eingeführten Aufgaben: (i) *Konfiguration der Modellparameter*, (ii) *Modellausführung*, (iii) *Datenakquisition*, (iv) *Datenanalyse* und (v) *Bewertung* herangezogen werden. Diese Aufgaben wurden ursprünglich für das Experimentieren mit nur einem Simulationsmodell eingeführt und müssen für die angestrebte Experimentdurchführung auf einem höheren Abstraktionsniveau zunächst angepasst werden. Abbildung 34 zeigt eine mögliche Experimentsteuerung mit den angepassten Aufgaben, welche nachfolgend beschrieben werden.

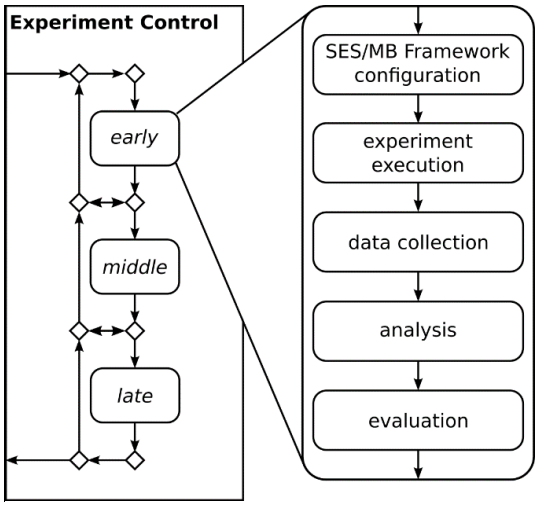


Abbildung 34: Schematische Darstellung einer Experimentsteuerung.

Konfiguration des SES/MB Frameworks

Diese Aufgabe befasst sich mit der Konfiguration des SES/MB Frameworks. Gemäß Gleichung 14 oder 15 müssen die notwendigen SES-Variablen, bezogen auf die Experimentphase, ausgewählt und mit zulässigen Werten belegt werden. Darauf aufbauend werden unter Nutzung der Methode *pruning* eine oder mehrere Experimentvarianten generiert und abgespeichert.

Experimentausführung

Es werden die generierten Experimentvarianten ausgeführt. In Abhängigkeit von der Ausführungseinheit können die Experimentvarianten *sequentiell* oder *parallel/verteilt*

ausgeführt werden. Eine parallele/verteilte Ausführung kann die Experimentausführungszeit reduzieren.

Datenakquisition

Diese Aufgabe beschreibt, welche Experimentergebnisse jeder Experimentvariante für die Datenauswertung auszuwählen sind, wie und ob die Daten einer Aufbereitung unterzogen werden müssen.

Datenanalyse

Es werden die akquirierten Daten analysiert. Im Rahmen des Screenings oder einer Sensitivitätsanalyse können die Ergebnisse auf die Signifikanz der einzelnen Parameter des Simulationsmodells untersucht werden und damit der entsprechende Parameterraum der zugehörigen Simulationsmodelle eingeschränkt werden. Bei einer Optimierung kann eine Analyse der berechneten optimalen Parameterkonfiguration erfolgen.

Bewertung

Bei der Bewertung werden die gesammelten Experimentergebnisse aller Experimentvarianten beispielsweise auf Plausibilität und hinreichende Gültigkeit sowie auf das zu erreichende Ziel überprüft. Sind die Ergebnisse zufriedenstellend, so kann die nächste Experimentphase starten oder das Experiment finalisiert werden. Im anderen Fall kann beispielsweise das aktuelle Experiment wiederholt oder zu einer vorherigen Experimentphase zurückgesprungen werden.

Der prinzipielle Ablauf eines automatisierten Experiments gemäß dem eingeführten Konzeptrahmen wird im nächsten Unterabschnitt erläutert.

5.1.2 Prinzipieller Ablauf eines automatisierten Experiments

Abbildung 35 zeigt den prinzipiellen Ablauf eines automatisierten Experiments in Form eines Sequenzdiagramms. Er ist für alle drei Experimentphasen identisch. Zu Beginn erfolgt die Initialisierung (*set(...)*) der drei Komponenten: (i) Experimentsteuerung, (ii) SES/MB Framework und (iii) Experimentausführungseinheit. Danach wird die Experimentsteuerung gestartet (*execute()*). Die *Experimentsteuerung* bringt die in Unterabschnitt 5.1.1 diskutierten Aufgaben sequenziell zur Ausführung. Als erstes wird die Konfiguration durchgeführt. Hierzu werden gemäß Gleichung 14 oder 15 entsprechende Konfigurationen der SES-Variablen (SESVars) erstellt und an das SES/MB Framework (*configuration(SESVars)*) übergeben. Dieses generiert für jede zulässige Konfiguration von SES-Variablen eine Experimentvariante (E) und gibt diese zurück an die Experimentsteuerung. Dann folgt die Experimentausführung durch Übergabe der generierten Experimentvarianten an die Experimentausführungseinheit (*execution(E)*). Je nach Art und Einstellung der Experimentausführungseinheit erfolgt eine sequentielle oder parallele Experimentausführung. Die prinzipielle Ausführung eines komplexen Experimentes erfolgt gemäß Abbildung 24 im Unterabschnitt 4.2.1. Ausführungen zur parallelen Abarbeitung erfolgen im nächsten Unterabschnitt (vgl. Abbildung 38).

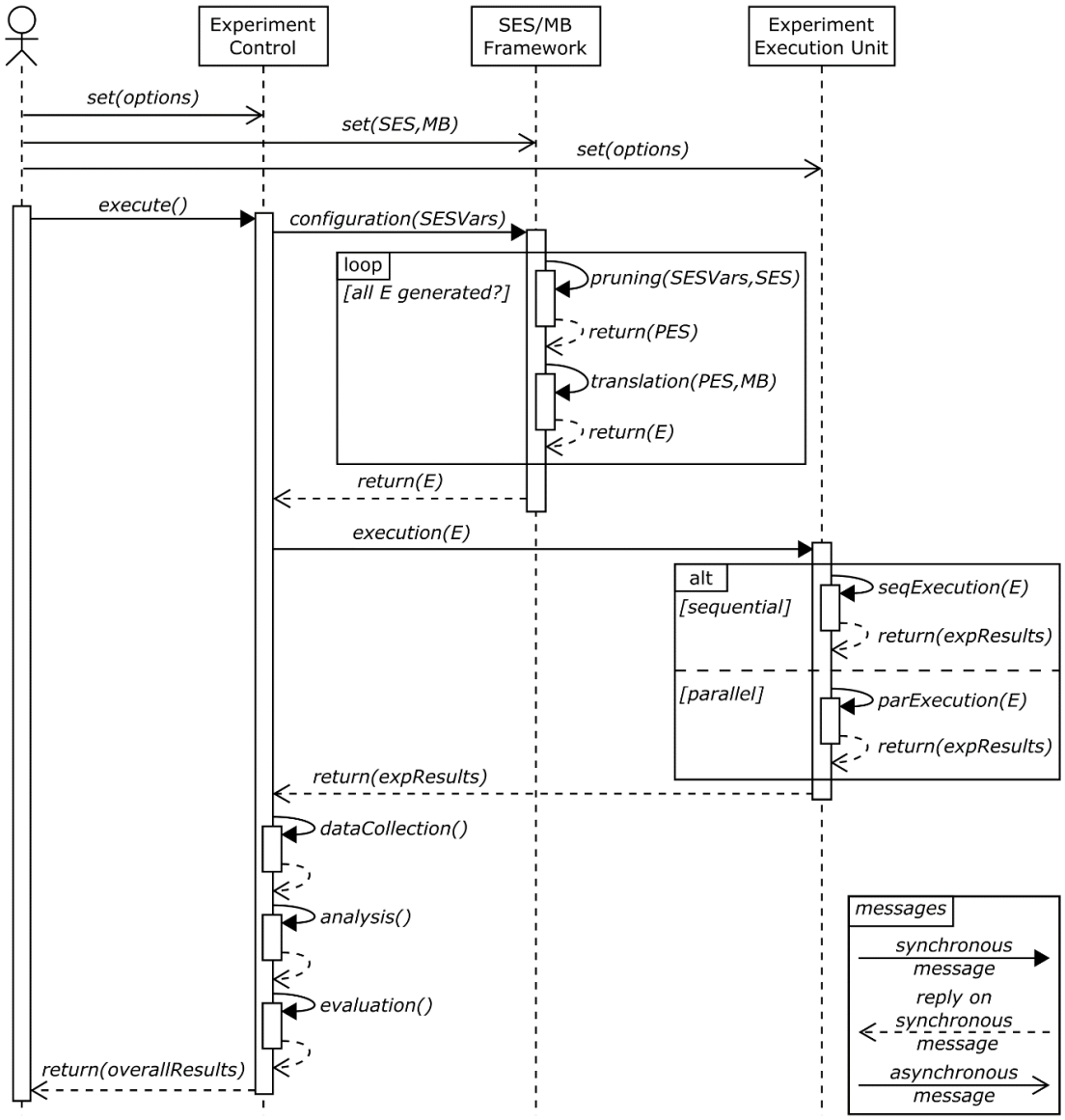


Abbildung 35: Prinzipieller Ablauf eines automatisierten Experiments auf Basis des Konzeptrahmens in Abbildung 33.

Ist die Experimentausführung abgeschlossen, werden die Ergebnisse der Experimentvarianten an die Experimentsteuerung zurückgegeben. Im nächsten Schritt, der Datenakquisition (*dataCollection()*), erfolgt eine Auswahl sowie Aufbereitung der gewünschten Datensätze für die sich anschließende Datenanalyse (*analysis()*). Hier werden beispielsweise die Experimentergebnisse jeder Experimentvariante vom Typ Screening auf signifikante Parameterkonfigurationen analysiert, um damit den Faktorraum eines untersuchten Simulationsmodells einzuschränken. Abschließend erfolgt die Bewertung (*evaluation()*) der Ergebnisse der Datenanalyse und es wird das Gesamtergebnis der Untersuchung durch die Experimentsteuerung an die übergeordnete Ebene gegeben. Diese entscheidet, ob ein Experiment wiederholt, abgebrochen oder eine nächste Experimentphase gestartet wird.

Bei umfangreichen Problemstellungen können schnell viele oder rechenzeitintensive Experimentvarianten anfallen. Dadurch kann gemäß Gehlsen [49] für eine durchzuführende Untersuchung oft erhebliche Ausführungszeit entstehen. Zur Reduktion der Ausführungszeiten kann eine parallele/verteilte Abarbeitung von Experimentvarianten beitragen. Dabei

werden die einzelnen Experimentvarianten lokal auf einem Mehrkernprozessor oder unter Einbindung entfernter Ressourcen verteilt durchgeführt. Im nachfolgenden Unterabschnitt erfolgen hierzu grundlegende Betrachtungen.

5.1.3 Betrachtungen zur verteilten und parallelen Experimentausführung

Zur Reduktion der Rechenzeit kann eine Experimentausführungseinheit mit Möglichkeiten einer parallele/verteilten Experimentausführung, zum Beispiel auf einem Mehrprozessorrechner oder Rechen-Cluster, beitragen. Nach der Aussage von Pawletta [113] handelt es sich bei der parallelen und verteilten Verarbeitung um verschiedene Teilgebiete der Informatik oder um ein grundlegendes Prinzip der rechnergestützten Problemlösung und es existiert keine allgemein akzeptierte Klassifikation. Bei den hier zu betrachtenden Problemstellungen kann pragmatisch auf die Parallelisierungsebenen in Fink [41] und Fink et al. [40] aufgebaut werden. Diese sind in Abbildung 36 beginnend mit der höchsten Ebene (links) dargestellt.

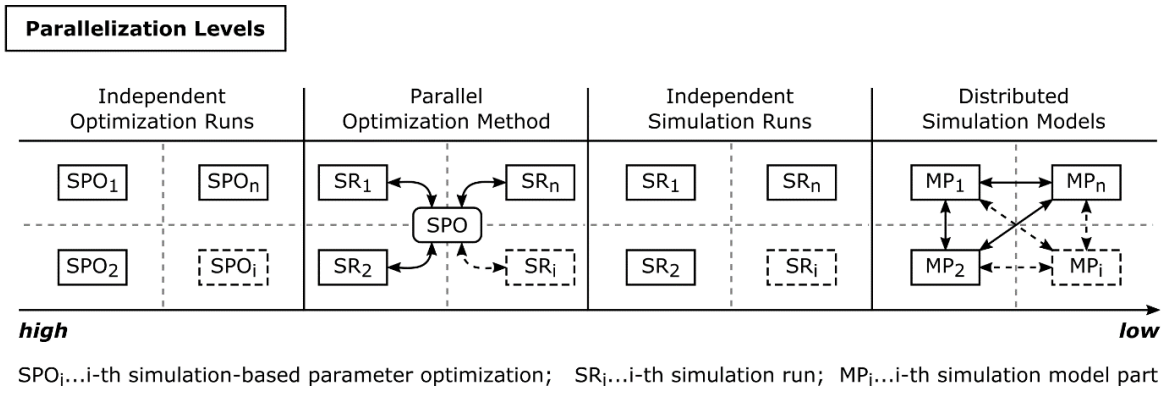


Abbildung 36: Schematische Darstellung der Parallelisierungsebenen basierend auf Recherchen in Fink [41] und Fink et al. [40].

Unabhängige Optimierungsläufe

Die höchste Ebene stellen *unabhängige Optimierungsläufe* dar. Dazu zählen Experimente mit unabhängigen Läufen simulationsbasierter Parameteroptimierungen (SPO). Zum Beispiel, wenn:

- eine Menge unterschiedlicher Simulationsmodelle einer SPO unterzogen werden,
- ein Simulationsmodell einer SPO mit unterschiedlichen Optimierungsverfahren unterzogen wird oder
- ein Simulationsmodell einer SPO mit unterschiedlichen Konfigurationen des Optimierungsverfahrens unterzogen wird.

Hierbei erhält jeder Prozessor eines Mehrprozessorrechners eine SPO als eine separate Einheit und die dafür notwendigen Simulationsläufe werden auf dem jeweiligen Prozessor sequenziell ausgeführt. Schulz [145] bezeichnet diesen Ansatz als *verteiltes Experimentieren*, da während der Abarbeitung keine Synchronisation zwischen den Prozessoren notwendig ist.

Parallele Optimierungsmethoden

Die Parallelisierungsebene *parallele Optimierungsmethoden* charakterisiert eine Optimierungsmethode mit der Möglichkeit der parallelen Ausführung der Zielfunktion. Im Kontext der SPO hängt die Zielfunktion eng mit den Simulationsergebnissen zusammen. Demnach zählt zu dieser Ebene eine SPO, bei welcher eine parallele Ausführung von Simulationsläufen im Rahmen der Zielfunktionsberechnung erfolgt. Beispiele für eine solche SPO sind der *Master-Slave Parallel Genetic Algorithm* von Köchel und Riedel [82] sowie Luque und Alba [98] oder der *Global Parallelization Genetic Algorithm* nach Alba und Troya [1]. Im Gegensatz zur vorherigen Parallelisierungsebene besteht ein höherer Kommunikationsaufwand zwischen den Prozessoren.

Unabhängige Simulationsläufe

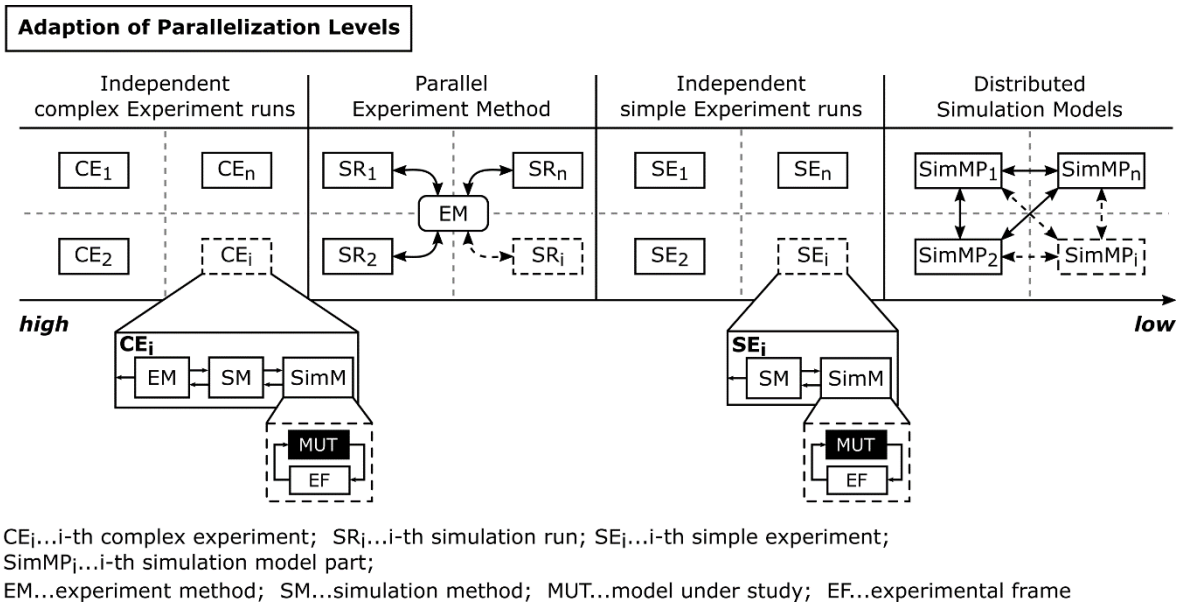
Die Ebene *unabhängige Simulationsläufe* besitzt Ähnlichkeiten zur Ebene *unabhängige Optimierungsläufe*. Der Unterschied besteht darin, dass hier ausschließlich isolierte Simulationsläufe den jeweiligen Prozessoren zugeordnet werden. Damit ist auch hier während der Abarbeitung keine Synchronisation zwischen den Prozessoren notwendig. Demnach entspricht diese Ebene den verteilten Experimenten nach Schulz [145]. Anwendungen für diese Ebene finden sich bei der stochastischen M&S im Sinne der *Multiple Replications in Parallel* nach Pawlikowski et al. [118] oder der *Parallel Independent Replication Simulation* nach Ndih und Charkaoui [166]. Genauso kann die von Nunes und Antunes [109] vorgestellte parallele explorative Analyse des Parameterraumes eines Simulationsmodells dieser Ebene zugeordnet werden.

Verteilte Simulationsmodelle

Die niedrigste Ebene bilden *verteilte Simulationsmodelle*. Im vereinfachten Sinne wird ein Simulationsmodell in mehrere Komponenten, meist als *Logische Prozesse* (LP) [145, 47] bezeichnet, partitioniert. Die einzelnen LPs werden während eines Simulationslaufes parallel durch einen Simulator oder mehrere Simulatoren abgearbeitet und kommunizieren über definierte Schnittstellen. Im Kontext der ereignisdiskreten M&S spricht beispielsweise Fujimoto [47, 48] von *Parallel and Distributed Discrete Event Simulation* oder Kunz [86], Schulz [145] sowie Ndih und Charkaoui [106] von *Parallel Discrete Event Simulation*. Wie in der Abbildung 36 angedeutet, ist bei diesen Methoden mit einem höheren Kommunikationsaufwand zwischen den einzelnen LPs zu rechnen. Nach Ndih und Cherkaoui [106] entsteht ein zusätzlicher Rechenaufwand zur Bewältigung zeitgetriebener oder ereignisgetriebener Synchronisation zwischen den LPs. Problemabhängig kann sich die Rechenzeit aufgrund des Kommunikationsaufwandes sogar verlängern.

Bezugnehmend auf die Klassifikation in Tabelle 2 im Abschnitt 2.2 sind die ersten beiden Parallelisierungsebenen in Abbildung 36, welche sich auf simulationsbasierte Parameteroptimierungen beziehen, relevant für die *Experimentphase langfristig* und sind bezogen auf den Experimentaufbau anwendbar auf *komplexe Experimente*. Die Parallelisierungsebene der unabhängigen Simulationsläufe steht in Verbindung mit der explorativen Analyse, welche sich in die *Experimentphase mittelfristig* einordnet und vom Experimentaufbau zu den *einfachen Experimenten* zählt.

Die Parallelisierungsebenen in Abbildung 36 geben keinen Bezug zu den Experimenten der *Experimentphasen kurzfristig* und *mittelfristig* im Zusammenhang mit dem Experimentaufbau *komplex*, besser gesagt sie nehmen keinen Bezug auf das simulationsbasierte Screening oder Sensitivitätsanalysen. Nachfolgend erfolgt ein Versuch einer weiteren Abstraktion der Parallelisierungsebenen zur besseren Anpassung an die in dieser Arbeit betrachteten Klassen von Experimentphasen und Experimentaufbauten. Abbildung 37 illustriert die Adaption der Parallelisierungsebenen.



CEi...i-th complex experiment; SRi...i-th simulation run; SEi...i-th simple experiment;
SimMPi...i-th simulation model part;
EM...experiment method; SM...simulation method; MUT...model under study; EF...experimental frame

Abbildung 37: Adaption der Parallelisierungsebenen aus Abbildung 36 auf die Experimentklassifikation in dieser Arbeit.

Im Anschluss werden die adaptierten Parallelisierungsebenen erläutert, wobei nur auf die wesentlichen Unterschiede eingegangen wird.

Unabhängige komplexe Experimentläufe

Die höchste Ebene entspricht der Ausführung von unabhängigen Experimentvarianten. Wie bereits in Unterabschnitt 2.2.2 und Abschnitt 4.2 diskutiert, zählen zu den komplexen Experimenten neben der Experimentmethode Parameteroptimierung auch das Screening oder Sensitivitätsanalysen. Analog der Ebene unabhängiger Optimierungsläufe in Abbildung 36 können auch unabhängige Screening- oder Sensitivitätsanalyse-Experimente auf mehreren Prozessoren separat parallel ausgeführt werden. Die jeweils zugehörigen Simulationsläufe werden auf den jeweiligen Prozessoren sequentiell ausgeführt.

Parallele Experimentmethoden

Auf dieser Ebene werden die untergeordneten Simulationsläufe einer Experimentmethode vom Typ Parameteroptimierung, Screening oder Sensitivitätsanalyse parallel ausgeführt.

Unabhängige einfache Experimentausführung

Diese Ebene steht für die parallele Ausführung unabhängiger Varianten einfacher Experimente. In Anlehnung an Unterabschnitt 3.2.2 können einfache Experimente einzelne Simulationsläufe oder mehrere Simulationsläufe im Sinne einer explorativen Analyse kodieren.

Verteilte Simulationsmodelle

Die Ebene entspricht den verteilten Simulationsmodellen in Abbildung 36. Der Unterschied besteht nur darin, dass das Simulationsmodell gemäß den Ausführungen in Abschnitt 4.1 umzusetzen ist.

Bezugnehmend auf den Konzeptrahmen für das automatisierte Experimentieren in Abbildung 33 erfolgt die weitere Betrachtung der parallelen Experimentausführung ausschließlich für die Parallelisierungsebenen *unabhängige komplexe Experimentläufe* und *unabhängige einfache Experimentläufe* gemäß Abbildung 37. Diese Entscheidung wird mit Verweis auf Gehlsen [49] und Schulz [145] begründet. Demnach lassen sich unabhängige Experimentläufe, von Schultz [145] verteiltes Experimentieren genannt, einfach parallelisieren, da keine Synchronisation der einzelnen Prozessoren oder Rechner notwendig ist.

Dustdar et al. [33] nennen die Master-Slave-Architektur als das am meisten verwendete Muster für die Parallelverarbeitung. Diese ist nach Schäuffele und Zurawka [136] einfach zu realisieren. Der Master zerlegt die zu bewältigende Aufgabe in semantisch identische Teilaufgaben und delegiert sie an mehrere ihm unterstellte Slaves. Letztere berechnen Einzelergebnisse und übermitteln diese an den Master, welcher die Einzelergebnisse zu einem Gesamtergebnis verknüpft.

Übertragen auf die vorliegende Problemstellung erhält der Master die Menge auszuführender Experimentvarianten, zerlegt diese in unabhängige Experimente und delegiert sie zur Ausführung an die unterstellten Slaves. Die Slaves führen das Experiment aus und übermitteln ihre Ergebnisse an den Master, der diese zu einem Gesamtergebnis verknüpft. Abbildung 38 stellt die Master-Slave-Architektur für diese Problemstellung schematisch dar.

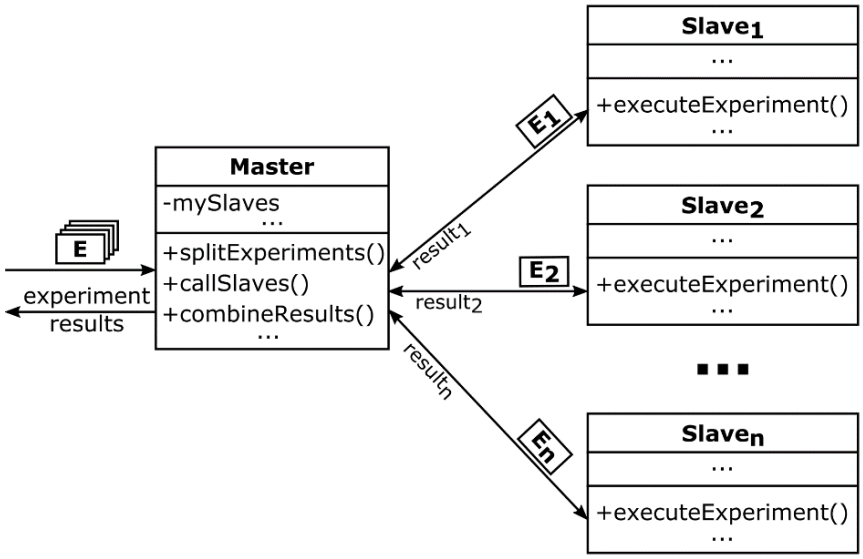


Abbildung 38: Schematische Darstellung der Master-Slave-Architektur nach [33] angepasst auf die spezielle Problemstellung.

Im nächsten Abschnitt wird der Einsatz des Konzeptrahmens aus Abbildung 33 im Sinne eines hochkomplexen Experiments, und zwar einer simulationsbasierten Struktur- und Parameteroptimierung, als eine alternative zum Ansatz aus dem Unterabschnitt 2.2.2 diskutiert.

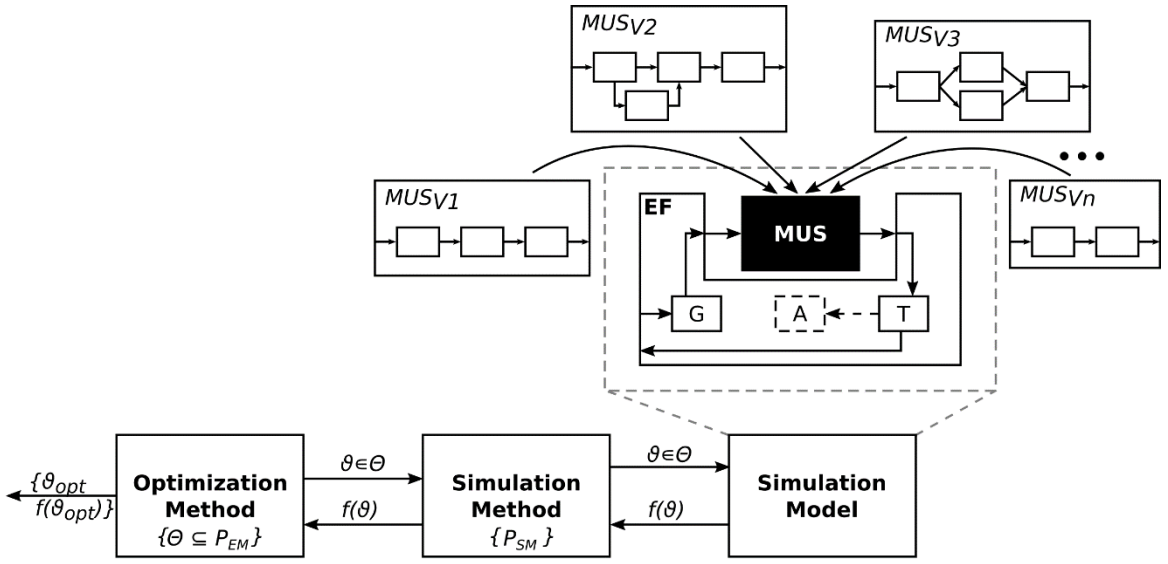
5.2 Lösungsansatz für hochkomplexe Experimente am Beispiel einer simulationsbasierten Struktur- und Parameteroptimierung (SSPO)

Gemäß der Klassifikation im Unterabschnitt 2.2.2 ist die simulationsbasierte Struktur- und Parameteroptimierung (SSPO) ein hochkomplexes Experiment. Durch die Optimierungsmethode werden nicht nur die Einflussgrößen des Simulationsmodells auf ein Zielfunktional untersucht, sondern auch der Einfluss unterschiedlicher Simulationsmodellstrukturen. Demnach besitzt der Suchraum Θ, wie bereits im Unterabschnitt 2.2.2 diskutiert, Informationen bezogen auf die Parametereinstellungen und die möglichen Strukturen eines Simulationsmodells. Das Ergebnis einer SSPO ist ein parameter- und strukturoptimiertes Simulationsmodell. Wie eine SSPO mit dem in Abbildung 33 vorgestellten Konzeptrahmen prinzipiell automatisiert werden kann, wird nachfolgend aufgezeigt. Dazu muss jedoch zuvor die Problemstellung der SSPO mit Bezug auf die in Kapitel 4 eingeführte Experimentstrukturierung betrachtet werden. Danach erfolgt die Diskussion zur Adaption des Konzeptrahmens und des daraus folgenden prinzipiellen Ablaufes einer SSPO. Abschließend wird ein vereinfachtes Beispiel vorgestellt.

5.2.1 Reanalyse der Problemstellung der SSPO

Im Rahmen des Variantenmanagements wurde der Begriff des Simulationsmodells im Kapitel 4 erweitert. Das Simulationsmodell besteht demnach aus einem Experimental Frame (EF), welcher den Kontext der zu untersuchenden Problemstellung codiert, und einem Model Under Study (MUS), einer abstrakten Repräsentation des zu untersuchenden Systems. Im Gegensatz dazu bildet das Simulationsmodell bei den Betrachtungen zur SSPO im Unterabschnitt 2.2.2 *ausschließlich* das MUS ab und das Computermodell in Abbildung 6 entspricht einem MUS mit einem zugehörigen Simulator.

Abbildung 39 zeigt die Anpassung eines SSPO Experiments an die im Abschnitt 4.2 eingeführte Experimentstruktur. Die Experimentmethode entspricht einer gewöhnlichen numerischen Optimierungsmethode. Letztere codiert den konkreten Optimierungsalgorithmus und beinhaltet eine Menge methodenspezifischer Parameter (P_{EM}) einschließlich des charakteristischen Suchraumes Θ. Die Simulationsmethode realisiert die Schnittstelle zwischen der Optimierungsmethode, dem Simulator sowie dem Simulationsmodell und spezifiziert, wie ein Simulationslauf konkret auszuführen ist. Dazu besitzt die Simulationsmethode methodenspezifische Parameter (P_{SM}). Jedes Simulationsmodell besteht wie bereits beschrieben aus einem MUS und dem EF. Der EF codiert den Kontext der zu untersuchenden Problemstellung, also die Zielfunktion wird im Transducer spezifiziert. Demgemäß berechnet dieser den Zielfunktionswert $f(\vartheta)$.



MUS...Model Under Study; MUS_{Vn}...n-th Variant of an MUS; EF...Experimental Frame;
G...Generator; A...Acceptor; T...Transducer

P_{EM}...parameters of experiment (optimization) method; Θ...search space;
P_{SM}...parameters of simulation method;

ϑ...configuration of optimization parameters; f(ϑ)...objective function value;
ϑ_{opt}... optimal parameter configuration; $f(\vartheta_{opt})$...objective function value at ϑ_{opt}

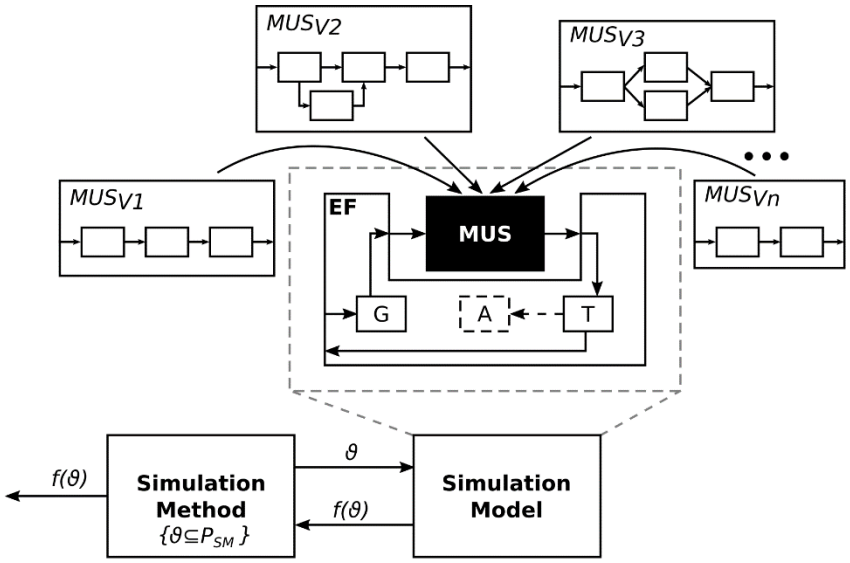
Abbildung 39: Adaption der SSPO gemäß der Strukturierung in Kapitel 4.

Im Rahmen der SSPO bezieht sich die Variabilität ausschließlich auf das MUS im Simulationsmodell. Da bei einer SSPO nicht nur die Modellparameter eines MUS den Untersuchungsgegenstand bilden, sondern auch eine Menge unterschiedlicher Modellstrukturen, müssen unterschiedliche MUS-Varianten mit unterschiedlichen Parametereinstellungen untersucht werden. Die Komponenten Optimierungsmethode, Simulationsmethode und der Experimental Frame sind für alle MUS-Varianten identisch. Demzufolge benötigt ein SSPO Experiment ausschließlich ein Variantenmanagement bezüglich der möglichen MUS und deren Ausführung durch eine Simulationsmethode. Bei einer Erweiterung der Problemstellung auf unterschiedliche Zielgrößen, wäre das Variantenmanagement zusätzlich auf eine Menge von EF zu erweitern.

Gemäß der Experimentstruktur der SSPO in Abbildung 39 müssen zur Berechnung eines Zielfunktionswertes $f(\vartheta)$ in einem Optimierungsschritt eine Menge von Simulationsmodellvarianten generiert und ausgeführt werden. Da die konkrete Ausführung eines Simulationslaufes in der Simulationsmethode codiert ist, muss diese beim Variantenmanagement mitbetrachtet werden. Nach Unterabschnitt 4.2.2 stellt die Ausführung eines Simulationslaufes ein *einfaches* Experiment dar. Abbildung 40 zeigt die Struktur dieser einfachen Experimente, wie sie im hochkomplexen Experiment der SSPO in jedem Optimierungsschritt auszuführen sind. Die von der Optimierungsmethode berechneten Parameter $\vartheta \in \Theta$ des aktuellen Optimierungsschrittes müssen als Teilmenge der methodenspezifischen Parameter (P_{SM}) der Simulationsmethode abgebildet werden.

Demgemäß sind bei der SSPO in jedem Optimierungsschritt die einzelnen Phasen des Variantenmanagements eines *einfachen Experiments* zu durchlaufen. Dazu müssen in einer SES alle Strukturvarianten des MUS, in Abbildung 40 mit $MUS_{V1} \ldots MUS_{Vn}$ bezeichnet,

sowie die zugehörigen Konfigurationsvarianten der Simulationsmethode spezifiziert werden. Weiterhin müssen in einer MB die elementaren Komponenten zur Generierung der ausführbaren Experimentvarianten organisiert werden.



MUS...Model Under Study; MUS_{Vn}...n-th Variant of an MUS; EF...Experimental Frame; G...Generator; A...Acceptor; T...Transducer

P_{SM}...parameters of simulation method;

ϑ...configuration of optimization parameters; f(ϑ)...objective function value;

Abbildung 40: Struktur der einfachen Experimente im hochkomplexen Experiment SSPO.

Die Adaption des entwickelten Konzeptrahmens in Abbildung 33 auf hochkomplexe Experimente vom Typ SSPO wird im nächsten Unterabschnitt vorgestellt.

5.2.2 Adaption des Konzeptrahmens für die SSPO

Die Adaption des Konzeptrahmens erfordert zunächst eine kurze Betrachtung zum Suchraum bei der SSPO im Kontext des SES/MB Frameworks. Der für die SSPO charakteristische Suchraum Θ wird gemäß Gleichung 13 in Abschnitt 5.1 über die Wertebereiche der SES-Variablen definiert, wie in Gleichung 16 gezeigt. Eine konkrete Wertebelegung der SES-Variablen entspricht damit einer spezifischen Konfiguration der Optimierungsparameter $\vartheta \in \Theta$.

$$\Theta = P_{SESVar} = P_{SESVar,1} \times P_{SESVar,2} \times \ldots \times P_{SESVar,n} \tag{16}$$

Nach Unterabschnitt 2.2.2 wird der Suchraum Θ bei einer SSOP durch die Vereinigung des klassischen Suchraums Θ_P auf Basis der konfigurierbaren Modellparameter eines Simulationsmodells mit der Menge aller möglichen Strukturen eines Simulationsmodell Θ_S gebildet. An dieser Stelle sei angemerkt, dass die SES-Variablen bisher ausschließlich im Kontext von Strukturvarianten benutzt wurden. Auf die Verwendung der SES-Variablen zur Abbildung konkreter Parameter im Sinne von Θ_P. Auf diesen Sachverhalt wird im nachfolgenden Unterabschnitt bei der Diskussion eines Beispiels noch explizit eingegangen.

Abbildung 41 zeigt die Adaption des Konzeptrahmens zum Aufbau eines SSPO-Experiments und Abbildung 42 zeigt den Ablauf in Form eines Sequenzdiagramms. Das

gesamte Framework besteht aus den Komponenten: (i) *Optimierungsmethode*, (ii) *Interfacemethod*e, (iii) *SES/MB Framework* und (iv) *Experimentausführungseinheit*. Gemäß der Reanalyse der Problemstellung der SSPO ist die Optimierungsmethode nicht Bestandteil der zu generierenden Experimente, sondern ein Teil der Experimentsteuerung in Bezug auf den Konzeptrahmen in Abbildung 33, genauso wie die neu eingeführte Interfacemethode. Letztere beschreibt die im Abschnitt 5.1 definierten Aufgaben der Experimentsteuerung und realisiert die Anbindung zwischen den einzelnen Komponenten. Die Variantengenerierung der in jedem Optimierungsschritt auszuführenden *einfachen Experimente*, also der *Simulationsläufe*, erfolgt weiterhin im SES/MB Framework und deren Ausführung in der Ausführungseinheit.

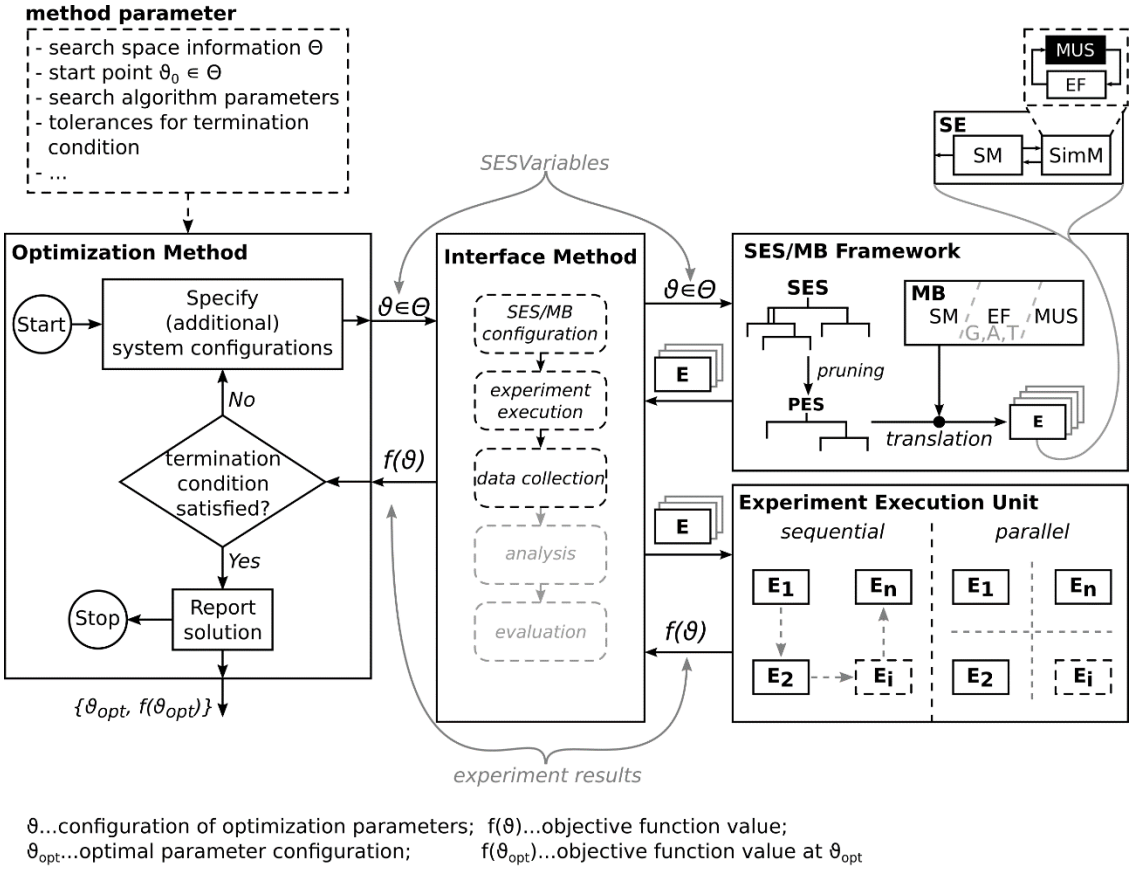


Abbildung 41: Adaption des Konzeprahmens aus Abbildung 33 für die Durchführung eines hochkomplexen Experimentes vom Typ SSPO.

Zu Beginn werden alle Komponenten initialisiert *(set(...))*. In diesem Zusammenhang sei nur erwähnt, dass der Suchraum Θ und ein Startpunkt $\vartheta_0 \in \Theta$ als methodenspezifische Parameter P_{EM} der Optimierungsmethode *(set(methParam))* übergeben werden. Das SSPO-Experiment wird über die Optimierungsmethode gestartet *(execute())*. Diese übergibt der Interfacemethode einen aktuellen Parametervektor *(execute(ϑ))*, welcher eine konkrete SES-Variablen-Konfiguration $P_{SESVar,c}$ repräsentiert. Die aktuellen Optimierungsparameter werden durch die Interfacemethode an das SES/MB Framework weitergeleitet *(execute(ϑ))*. Diese erzeugt die gewünschten Experimentvarianten *E*, die Simulationsläufe, vom Typ *einfaches Experiment* gemäß Abbildung 40 und gibt diese an die Interfacemethode zurück *(return(E))*. Die Interfacemethode transferiert die Experimente zur Abarbeitung an die

Experimentausführungseinheit *(execute(E))*. Diese liefert die Ergebnisse in Form eines Zielfunktionswertes $f(\vartheta)$ für jedes ausgeführte Experiment zurück. Wie in Abbildung 41 gezeigt, kann die Interfacemethode alle für eine Experimentsteuerung spezifizierten Aufgaben ausführen. Im einfachen Fall der sequentiellen Experimentausführung transferiert sie lediglich den aktuell berechneten Zielfunktionswert f(ϑ) an die Optimierungsmethode *(return(f(ϑ)))*. Diese prüft, ob eine Abbruchbedingung eingetreten ist. Wenn dies nicht der Fall ist wird eine neue Belegung der Optimierungsvariablen ϑ bestimmt und der Zyklus wiederholt. Anderenfalls wird die Optimierung mit Rückgabe der gefundenen optimalen Parameter ϑ_{opt} und dem zugehörigen Zielfunktionswert $f(\vartheta_{opt})$ beendet. Da ϑ_{opt} genau einer Konfiguration der SES-Variablen entspricht, könnte in einem nächsten Schritt durch das SES/MB Framework *(execute(ϑ_{opt}))* direkt das parameter- und strukturoptimierte Simulationsmodell generiert werden.

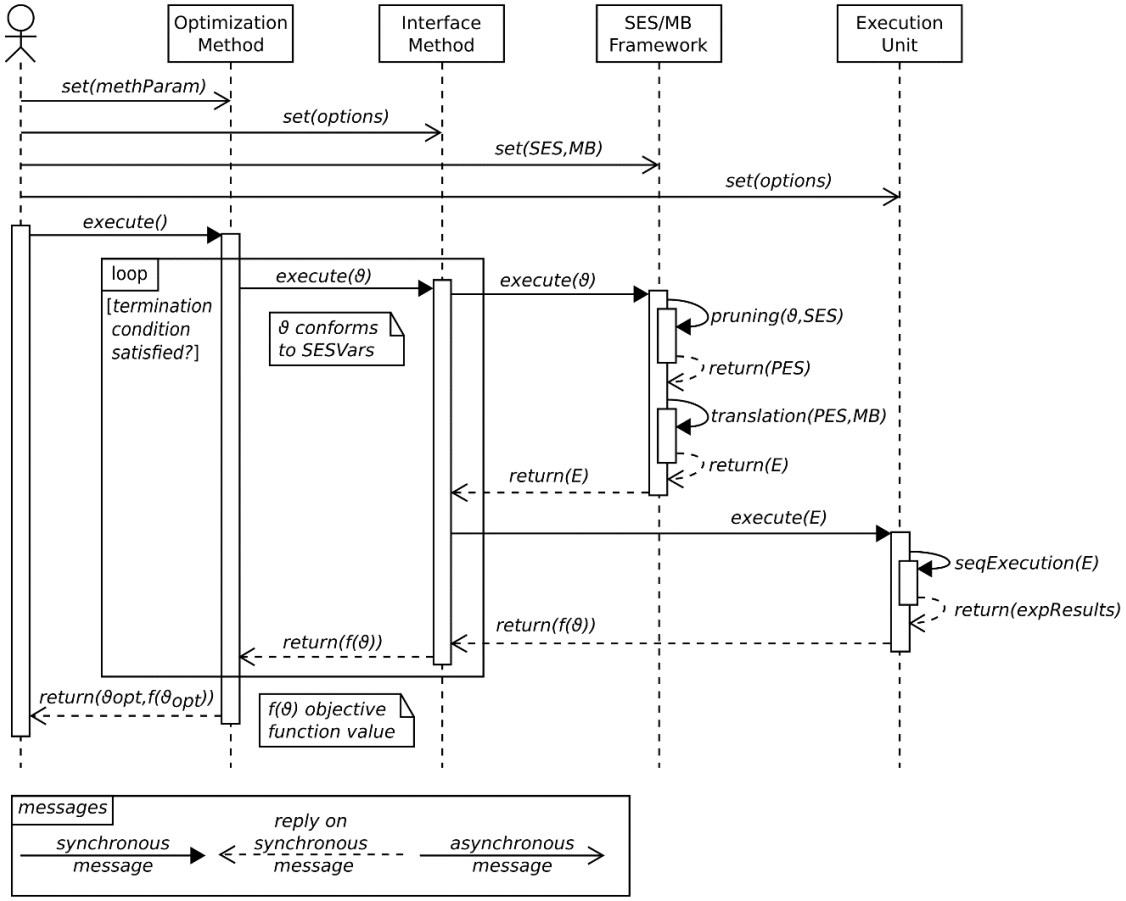


Abbildung 42: Ablauf eines hochkomplexen Experiments am Beispiel der SSPO mit der Einschränkung auf eine sequentielle Experimentausführung.

Im Anschluss wird die Spezifikation einer SES und der zugehörigen MB für eine SSPO an einem Beispiel demonstriert.

5.2.3 Beispiel zur Spezifikation einer SES und MB für eine SSPO

Für die Durchführung einer SSPO muss nach Unterabschnitt 5.2.1 ein Variantenmanagement vom Typ *einfaches Experiment* erfolgen. Die konkrete Experimentstruktur

dazu wurde in Abbildung 40 entwickelt. Abbildung 43 zeigt an einem Beispiel eine entsprechende Variantenspezifikation mit einer SES und einer MB.

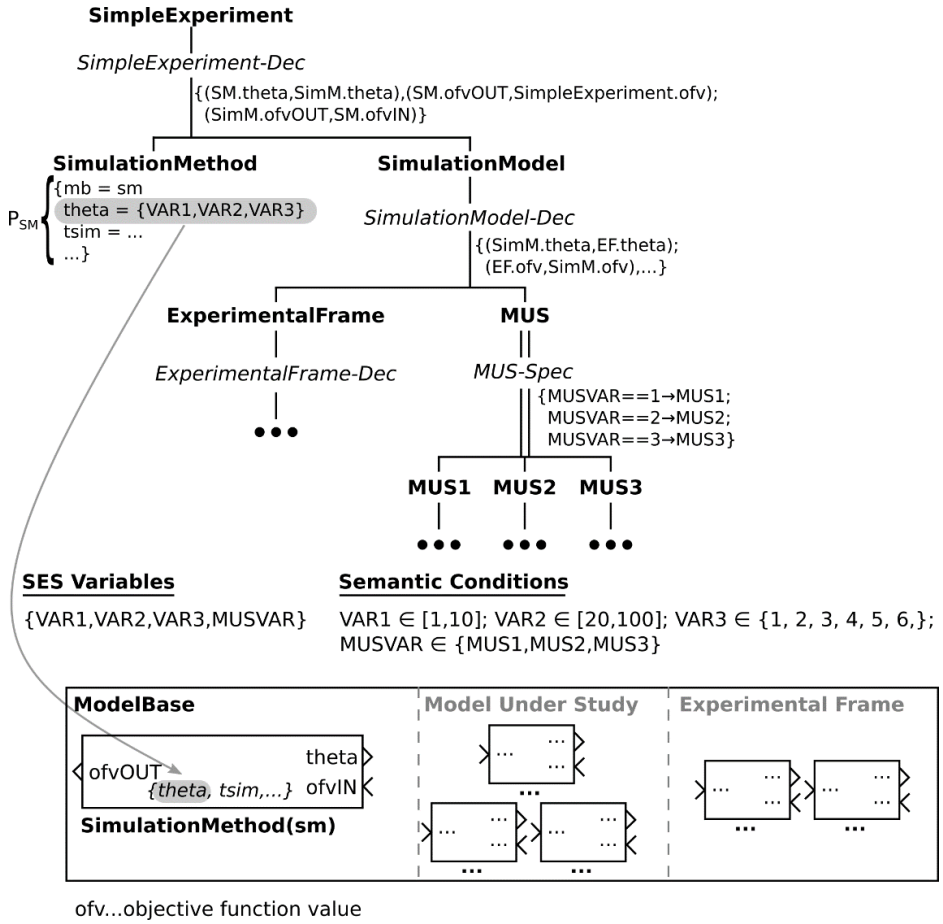


Abbildung 43: Spezifikation unterschiedlicher Varianten *einfacher* Experimente in einer SES mit einer schematisch angedeuteten MB für eine SSPO.

Mit dem Variationspunkt *MUS-Spec* im Teilbaum der Entität *SimulationModel* beschreibt die SES drei Strukturvarianten des Simulationsmodells. Weiterhin zeigt Abbildung 43 auch die Möglichkeit der Spezifikation von methodenspezifischen Parametern in Attributen, wie im Beispiel an der Entität *SimulationMethod*. Im Fall der SSPO müssen die zu optimierenden Modellparameter $\vartheta \in \Theta_P$ (vgl. Abb. 40) als methodenspezifische Parameter abgebildet werden. Da die Wertebelegung der ϑ variabel ist und für jeden Optimierungsschritt neu berechnet wird, erfolgt die Spezifikation mit SES-Variablen. Im dargestellten Fall werden die SES-Variablen *VAR1*, *VAR2* und *VAR3* dem Attribut *theta* zugewiesen. Ihre Wertebereiche, durch die der Parameterraum Θ_P definiert wird, sind in den *Semantic Conditions* beschrieben. Zusammenfassend zeigen die Gleichungen 17 bis 19 den mit der SES beschriebenen Suchraum Θ. Beim Ableiten einer konkreten Experimentvariante durch das *Pruning* erfolgen neben der Beschneidung des SES-Baumes auch die entsprechenden Wertzuweisungen an den Attributen.

$$\Theta_P = [1,10] \times [20,100] \times \{1,2,3,4,5,6\} \tag{17}$$

$$\Theta_S = \{MUS1, MUS2, MUS3\} \tag{18}$$

$$\Theta = \Theta_P \times \Theta_S \tag{19}$$

Es folgt eine kurze Zusammenfassung des Kapitels.

5.3 Zusammenfassung

Am Anfang des Kapitels wurde ein Konzeptrahmen für das automatisierte Experimentieren mit Experimentvarianten vom Typ *komplexes Experiment* vorgestellt. Dieser basiert auf den drei Komponenten: (i) Experimentsteuerung, (ii) SES/MB Framework und (ii) Experimentausführungseinheit. Die Experimentsteuerung beschreibt welche Experimente in welcher Reihenfolge zu generieren sind. Dabei kann sich die Reihenfolge von Experimenten auch reaktiv auf Basis aktueller Ergebnisse ergeben, was im Kapitel aber nicht weiter betrachtet wurde. Das SES/MB Framework codiert alle möglichen Experimentvarianten und generiert auf Anweisung der Experimentsteuerung ein oder eine Menge ausführbarer Experimente. Die Experimentausführungseinheit übernimmt die Experimentausführung. Da umfangreiche Untersuchungen schnell zu langen Rechenzeiten führen können, werden am Ende des ersten Abschnitts verschiedene Ansätze zur parallelen/verteilten Ausführung von Experimenten betrachtet.

Aufbauend auf dem entwickelten Konzeptrahmen erfolgte im zweiten Kapitelabschnitt eine Betrachtung zum automatisierten Experimentieren am Beispiel der simulationsbasierten Struktur- und Parameteroptimierung (SSPO), eine Untersuchung vom Typ *hochkomplexes Experiment*. Dabei zeigte sich, dass mit einer einfachen Adaption des Konzeptrahmens auch eine SSPO-Untersuchung automatisiert ausgeführt werden kann.

6 Anwendungsbeispiel

In diesem Kapitel wird die entwickelte Methode zum Variantenmanagement in der Modellbildung und Simulation auf Basis eines erweiterten SES/MB Frameworks an einem Beispiel aus dem Anwendungsbereich der Produktion und Logistik vorgestellt. Dabei stehen 81 unterschiedliche Prozesskettenvarianten zur Verfügung. Ziel ist die Identifikation einer optimalen Prozesskette für das Herstellen von rotationssymmetrischen Bauteilen. Im erweiterten Sinne entspricht eine fertigungstechnische Prozesskettenvariante mehreren aufeinanderfolgenden fertigungstechnischen Verfahren zur Herstellung eines Bauteils oder einer Bauteileigenschaft [105].

Die softwaretechnische Umsetzung erfolgt prototypisch in der Softwareumgebung MATLAB/Simulink [158]. Wie der Name andeutet besteht die Softwareumgebung allgemein aus zwei Grundkomponenten: (i) MATLAB [162] und (ii) Simulink [164]. Pawletta ordnet MATLAB allgemein den wissenschaftlich-technischen Entwicklungsumgebungen (engl. scientific and technical computing environment (SCE)) zu [113]. Das zentrale Anliegen der SCEs, im Gegensatz zu klassischen Programmiertechniken, liegt in der effektiven Prototypenentwicklung und ist durch zwei Kernkonzepte: (i) der interpretativen Abarbeitung und (ii) der array-orientierten Programmierung geprägt [113]. Die interpretative Abarbeitung ermöglicht die interaktive Ausführung von Routinen. Durch das Wegfallen der Kompilation können Routinen einfach und schnell getestet werden. Array-orientierte Sprachen, in MALAB M-language genannt [4], realisieren eine mathematiknahe Vektor- beziehungsweise Matrixnotation, wodurch numerische Berechnungen sehr effizient codiert werden können. Simulink hingegen ist eine weitverbreitete Model-Based-Design- und M&S-Umgebung. Die Modellierung von Systemen erfolgt unter Verwendung einer graphischen Modellierungssprache mittels Blockschaltbildern. Da sich das gewählte Anwendungsfeld der Produktion und Logistik in das ereignisdiskrete Modellierungsparadigma einordnen lässt, wird im Weiteren MATLAB/SimEvents als eine Erweiterung von Simulink verwendet. Abschließend wird darauf hingewiesen, dass keine Implementierungsdetails präsentiert und diskutiert werden.

6.1 Problembeschreibung

Beim Anwendungsbeispiel geht es um eine detaillierte, modellbasierte Untersuchung unterschiedlicher Prozessketten für eine Bauteilfertigung. Abbildung 44 zeigt schematisch ein Rohteil und das Fertigteil.

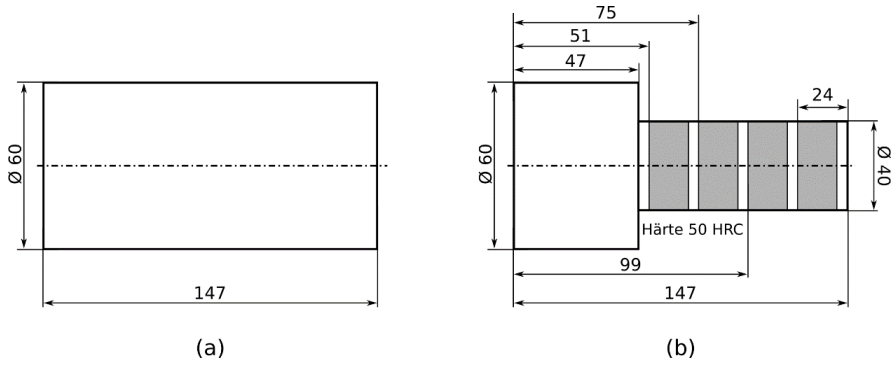
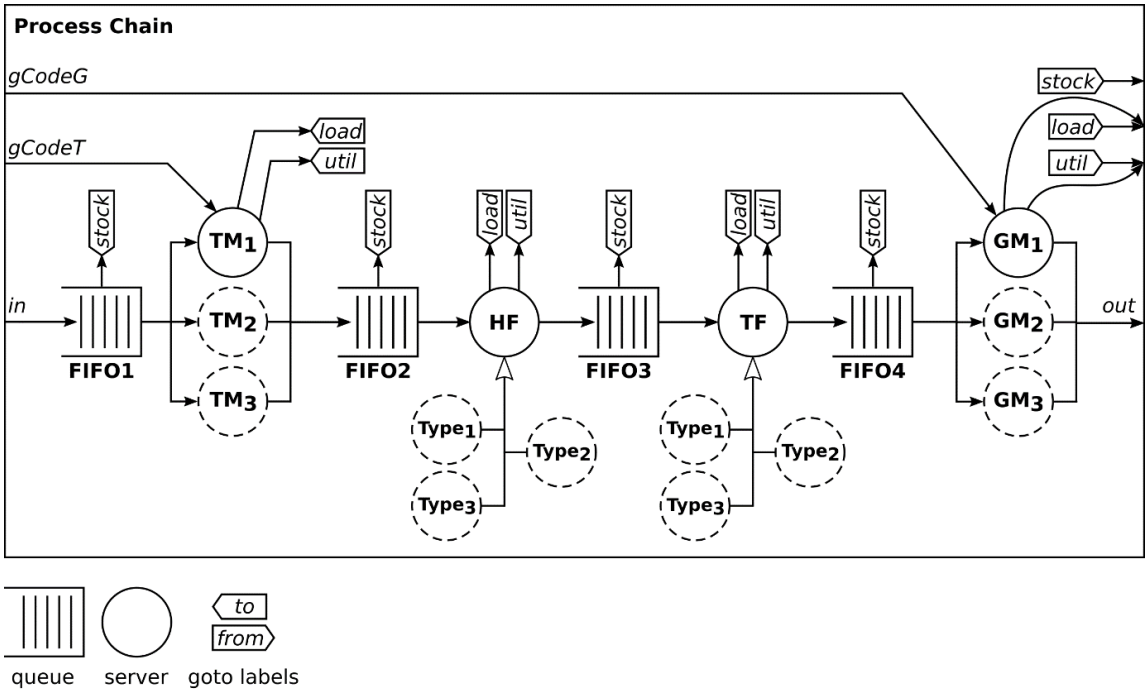


Abbildung 44: (a) Rohteil und (b) Fertigteil.

Die Grundidee der Prozesskette zur Fertigung des Bauteils entstammt aus Larek [87]. Sie besteht aus den vier Fertigungsoperationen (i) Außenlängsdrehen, (ii) Volumenhärten, (iii) Spanungsarmglühen (Anlassen) sowie (iv) Außenrundschleifen, wie in Abbildung 45 dargestellt. Auf der linken Seite (*in*) fließen Rohlinge in die Prozesskette hinein und auf der rechten Seite (*out*) verlassen die Fertigteile die Prozesskette. Auf Basis dieser Prozesskette soll im Rahmen einer simulationsbasierten Untersuchung eine Fertigungsstruktur, nachfolgend als Prozesskettenstruktur bezeichnet, entwickelt werden. Dabei gelten folgende Restriktionen. Für die Fertigungsoperationen Außenlängsdrehen und Außenrundschleifen stehen eine bis drei typgleiche CNC-Drehmaschinen ($TM_{1\text{-}3}$) sowie CNC-Schleifmaschinen ($GM_{1\text{-}3}$) zur Verfügung. Die Anzahl der zeitgleich eingesetzten TM und GM kann also zwischen eins und drei variieren. Für die Wärmebehandlungen (*HF*, *TF*) können drei unterschiedliche Ofentypen verwendet werden. Dabei gilt, dass für eine Wärmebehandlungsoperation jeweils nur ein Ofen beliebigen Typs eingesetzt werden darf. Die Unterschiede der einzelnen Ofentypen sind in der Tabelle 9 zusammengefasst. Zwischen den einzelnen Fertigungsschritten befinden sich FIFO-Puffer, deren Kapazität jeweils auf 400 Bauteile begrenzt ist. Unter Annahme dieser Bedingungen ergeben sich insgesamt 81 verschiedene Prozesskettenstrukturen für die Bauteilfertigung. Hinzu kommt, dass in jeder Prozesskettenstruktur die typgleichen CNC-Maschinen (TM und GM) in vorgegebenen Grenzen unterschiedlich parametriert werden können, wie nachfolgend noch betrachtet. Daraus folgt eine enorm große Anzahl möglicher Varianten zur Herstellung des Bauteils.



TMi...i-th turning machine; GMi...i-th grinding machine; TF...tempering furnace; HF...hardening furnace; util...utilization; gCodeT...g code for turning machine; gCodeG...g code for grinding machine; in...blank parts in; out...finished parts out

Abbildung 45: Schematische Darstellung der möglichen Prozesskettenvarianten zum Fertigen des Bauteils aus Abbildung 44.

Tabelle 9: Zusammenfassung der Unterschiede zwischen den Ofentypen.

	Typ1	Typ2	Typ3
Heizleistung [W]	6400	15000	20000
Kapazität [Stk]	20	40	70
Innenfläche [m^2]	0.14	0.25	0.35
Außenfläche [m^2]	3.6	4.8	6.15
Gewicht [kg]	80	240	370

Nachfolgend werden die Ein- und Ausgabegrößen der einzelnen Fertigungsoperationen und Einrichtungen näher betrachtet, um die variablen Einflussgrößen der Prozessketten-strukturen zu analysieren. Daran anschließend werden die Untersuchungsziele präzisiert.

Einflussgrößen

Alle CNC-Drehmaschinen erhalten als Eingabegrößen die Rohteile (*in*) und den für die Bearbeitung eines Rohteils anzuwendenden G-Code (*gCodeT*). Der G-Code entspricht in diesem Fall einer Funktion von der Schnitttiefe (*ap*), dem Vorschub (*f*) und der Schnittgeschwindigkeit (*vc*), wie in der Gleichung 20 verdeutlicht. Die Parameter des G-Codes können in vorgegebenen Grenzen variieren, wie in Tabelle 10 gezeigt. Somit bilden *ap*, *f*, und *vc* Einflussgrößen jeder Prozesskettenstruktur.

$$\mathrm{gCodeT(ap,f,vc)} \tag{20}$$

Die Wärmebehandlungsprozesse erhalten, genauso wie die Puffer, als Eingabegrößen nur die Bauteile. Die Parameter der verwendeten Ofentypen und die Pufferkapazitäten sind konstante Größen einer Prozesskettenstruktur und bilden damit keine zusätzlichen Einflussgrößen.

Die CNC-Schleifmaschinen erhalten als Eingabegrößen wärmebehandelte Bauteile sowie den zugehörigen G-Code (*gCodeG*). Analog zu den CNC-Drehmaschinen entspricht der gCodeG einer Funktion, wie in Gleichung 21 dargestellt. Dabei bezeichnen *vfr1…vfr3* die Vorschubgeschwindigkeiten für das Schruppen, Schlichten sowie Feinschlichten und *vc* entspricht der Schnittgeschwindigkeit. Auch die Parameter des gCodeG können in definierten Grenzen variiert werden und bilden damit Einflussgrößen der Prozesskettenstrukturen.

$$\mathrm{gCodeG(vfr1, vfr2, vfr3, vc)} \tag{21}$$

Zur Unterscheidung der Schnittgeschwindigkeiten in Gleichung 20 und 21 sind diese mit *vct (gCodeT)* beziehungsweise *vcg (gCodeG)* in der Tabelle 10 abgekürzt.

Tabelle 10: Zusammenfassung der Untersuchungsgrößen der G-Codes mit deren unteren (UG) und oberen Grenzwerten (OG).

	ap [mm]	f $\left[\frac{\text{mm}}{\text{U}}\right]$	vct $\left[\frac{\text{m}}{\text{min}}\right]$	vfr1 $\left[\frac{\text{mm}}{\text{min}}\right]$	vfr2 $\left[\frac{\text{mm}}{\text{min}}\right]$	vfr3 $\left[\frac{\text{mm}}{\text{min}}\right]$	vcg $\left[\frac{\text{m}}{\text{s}}\right]$
UG	0.5	0.1	150	0.48	0.19	0.03	30
OG	1.5	0.5	350	1.86	0.53	0.16	50

Ausgabegößen

Als Ausgabegrößen liefern alle Fertigungseinrichtungen und Puffer abgehende Bauteile sowie charakteristische Bewertungsgrößen. Die Puffer geben den Pufferbestand (*stock*) und die Fertigungseinrichtungen die Auslastung (*util*) sowie die elektrische Leistungsaufnahme (*load*) über der Zeit aus.

Untersuchungsziel

Das übergeordnete Untersuchungsziel ist die Ermittlung einer möglichst optimalen Prozesskettenvariante für ein Fertigungsszenario in einem definierten Untersuchungszeitraum. Für das betrachtete Beispiel sind das zwei Wochen. Das heißt, es ist eine Prozesskettenstruktur mit: Anzahl der CNC-Drehmaschinen und CNC-Schleifmaschinen, deren G-Code Parameter nach Tabelle 10 sowie der jeweilige Ofentyp für die beiden Wärmebehandlungsprozesse zu bestimmen. Für eine optimale Prozesskettenvariante gelten folgende Zielgrößenvorgaben:

1. Durchlaufzeit (*procTime*) in [h] minimal,
2. Durchsatz (*thrput*) in [Stk] maximal,
3. Energie pro Fertigteil (*Espec*) in $\left[\frac{\text{kWh}}{\text{Stk}}\right]$ minimal,
4. elektrische Lastspitze (*loadPeak*) in [kW] minimal,
5. Auslastung (*pcUtil*) in [%] maximal und
6. Bestand (*pcStock*) in [Stk] minimal.

Zur Bestimmung einer optimalen Prozesskettenvariante müssen die verschiedenen Prozesskettenstrukturen mit unterschiedlichen Parametrierungen der variablen Einflussgrößen des G-Codes untersucht werden. Dazu ist für jede Prozesskettenstruktur ein Simulationsmodell mit spezifischen konstanten Parametern (z.B. den typabhängigen Ofenparametern) zu generieren und damit ist gemäß Tabelle 10 der Parameterraum nach Gleichung 22 zu untersuchen.

$$P = (0.5,1.5) \times (0.1,0.5) \times (150,350) \times (0.48,1,86) \times \ldots \\ (0.19,0.53) \times (0.03,0.16) \times (30,50) \tag{22}$$

Um die Komplexität der Untersuchung zu beherrschen, ist es notwendig, so früh wie möglich nicht-zielführende Prozesskettenvarianten auszuschließen. Dazu soll jede Prozesskettenvariante gemäß den übergeordneten Experimentzielen (i) Screening, (ii) Sensitivitätsanalyse und (iii) Parameteroptimierung untersucht werden. Die Parameteroptimierung einer Prozesskettenstruktur erfolgt nur bezüglich der verbliebenen signifikanten Parameter, also für einen Suchraum $\Theta \subseteq P$. Die Zielfunktion der Optimierung ist als gewichtete Funktion

gemäß Gleichung 23 definiert und verknüpft die einzelnen Zielgrößen zu einem Zielfunktionswert.

$$\min_{\vartheta \in \Theta} f(\vartheta) = \frac{1}{6} procTime - \frac{1}{6} thrput + \frac{1}{6} Espec + \frac{1}{6} loadPeak - \frac{1}{6} pcUtil + \frac{1}{6} pcStock \quad (23)$$

Die Vorzeichen der einzelnen Terme geben Aufschluss über die Minimierung (+) oder Maximierung (-) der Zielgröße. Im gezeigten Fall besitzt jede Zielgröße den gleichen Wichtungsfaktor von $\frac{1}{6}$.

6.2 Variantenanalyse und konzeptuelle Modellierung

Entsprechend der Problembeschreibung erfolgen in diesem Abschnitt die Variantenanalyse und die konzeptuelle Modellierung. Gemäß Kapitel fünf müssen die übergeordneten Komponenten eines komplexen Experiments: (i) das Simulationsmodell, bestehend aus dem Model Under Study (MUS) und dem Experimental Frame (EF), (ii) die Simulationsmethode und (iii) die Experimentmethoden, herausgearbeitet werden. Darauf aufbauend, sind für das automatisierte Experimentieren eine SES und eine MB mit konfigurierbaren Simulationsmodellkomponenten, kurz Komponenten, zu entwickeln.

6.2.1 Model Under Study

Der konzeptuelle Aufbau des MUS, inklusive der strukturellen Variabilität, ist in Abbildung 46 gezeigt. Alle Prozesskettenvarianten besitzen die gleichen Einflussgrößen (IM) und Ausgabegößen (O_M). Die strukturelle Variabilität des MUS ist auf die Komponenten *TMs* und *GMs* begrenzt. Diese können aus einem, zwei oder drei typgleichen *TM* beziehungsweise *GM* Komponenten bestehen. Gleichzeitig variieren mit der Anzahl der *TM* und *GM* Komponenten die zugehörigen Kopplungsbeziehungen. Aus der Struktur des MUS in Abbildung 46 können, die in einer MB zu organisierenden, dynamischen Komponenten identifiziert werden. Wie in der Abbildung 47 zu sehen ist, enthält die MB nur vier Komponenten. Nach der Problembeschreibung sind die *TM*, die *GM* und die Puffer *FIFO1-4* typgleich. Das bedeutet, dass zur Komposition eines MUS die Komponenten *TM, GM* und *FIFO* mehrfach instanziiert und wenn notwendig, spezifisch parametriert werden. An dieser Stelle sei daran erinnert, dass die Parametrierung der *TM* und *GM* Komponenten über Eingangsvariablen erfolgt, da die G-Code Parameter variable Untersuchungsgrößen jeder Prozesskettenstruktur darstellen. Die drei Ofentypen sind hinsichtlich ihrer Dynamikbeschreibung identisch und unterscheiden sich ebenfalls nur in der Parametrierung. Damit können auch die Wärmebehandlungsoperationen (*HF, TF*) durch Instanzen der Komponente *HeatTreatment* und spezifischer Parametrierung im MUS abgebildet werden.

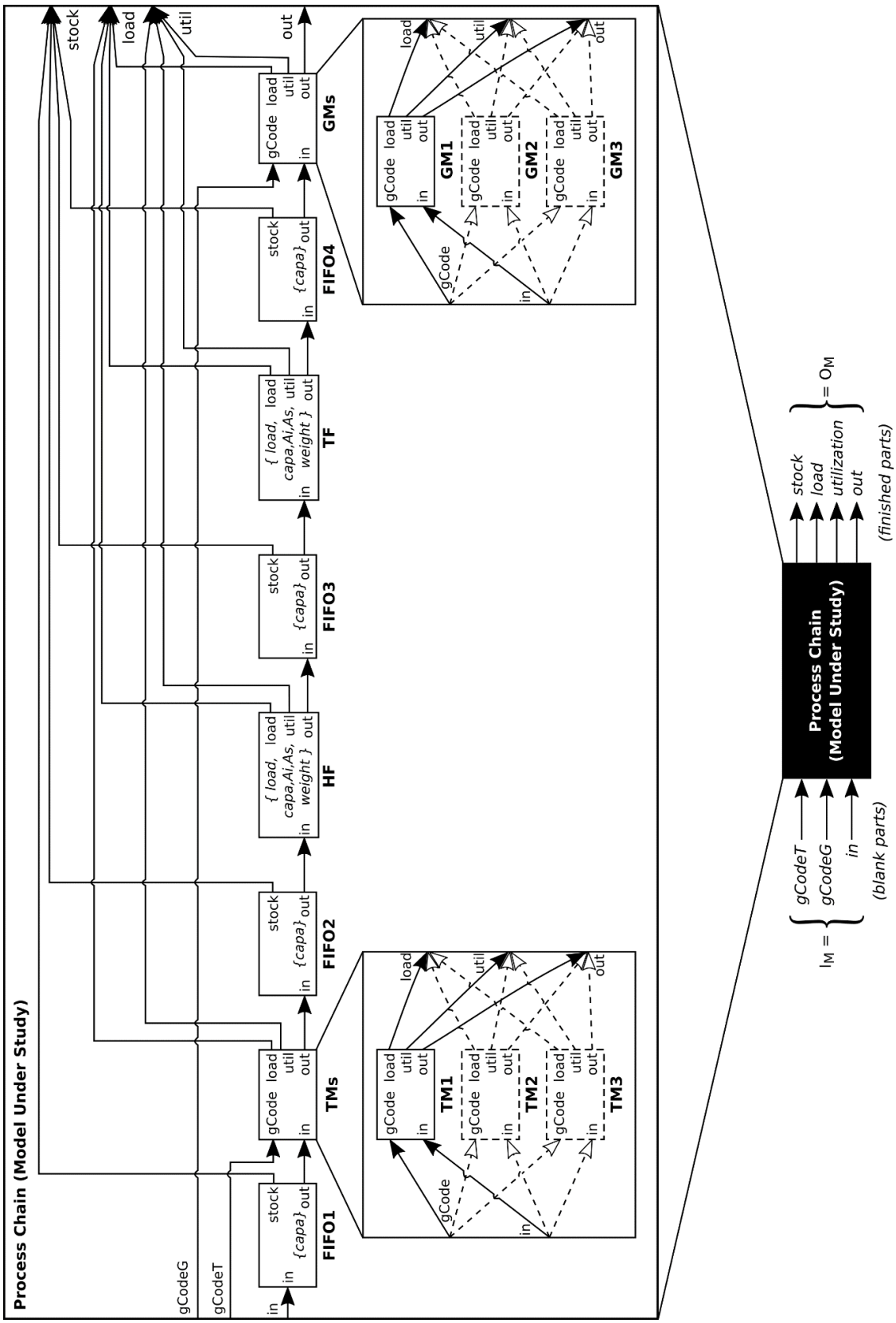


Abbildung 46: Konzeptuelles Modell des Modell Under Study einschließlich struktureller Variabilität.

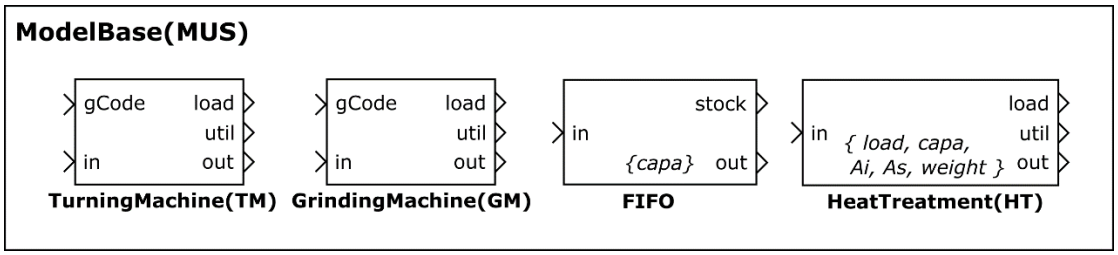


Abbildung 47: Organisation der MUS Komponenten in einer Modellbibliothek (MB).

Basierend auf dem konzeptuellen Modell in Abbildung 46 ist die Menge der MUS Strukturvarianten vollständig in der SES in Abbildung 48 modelliert.

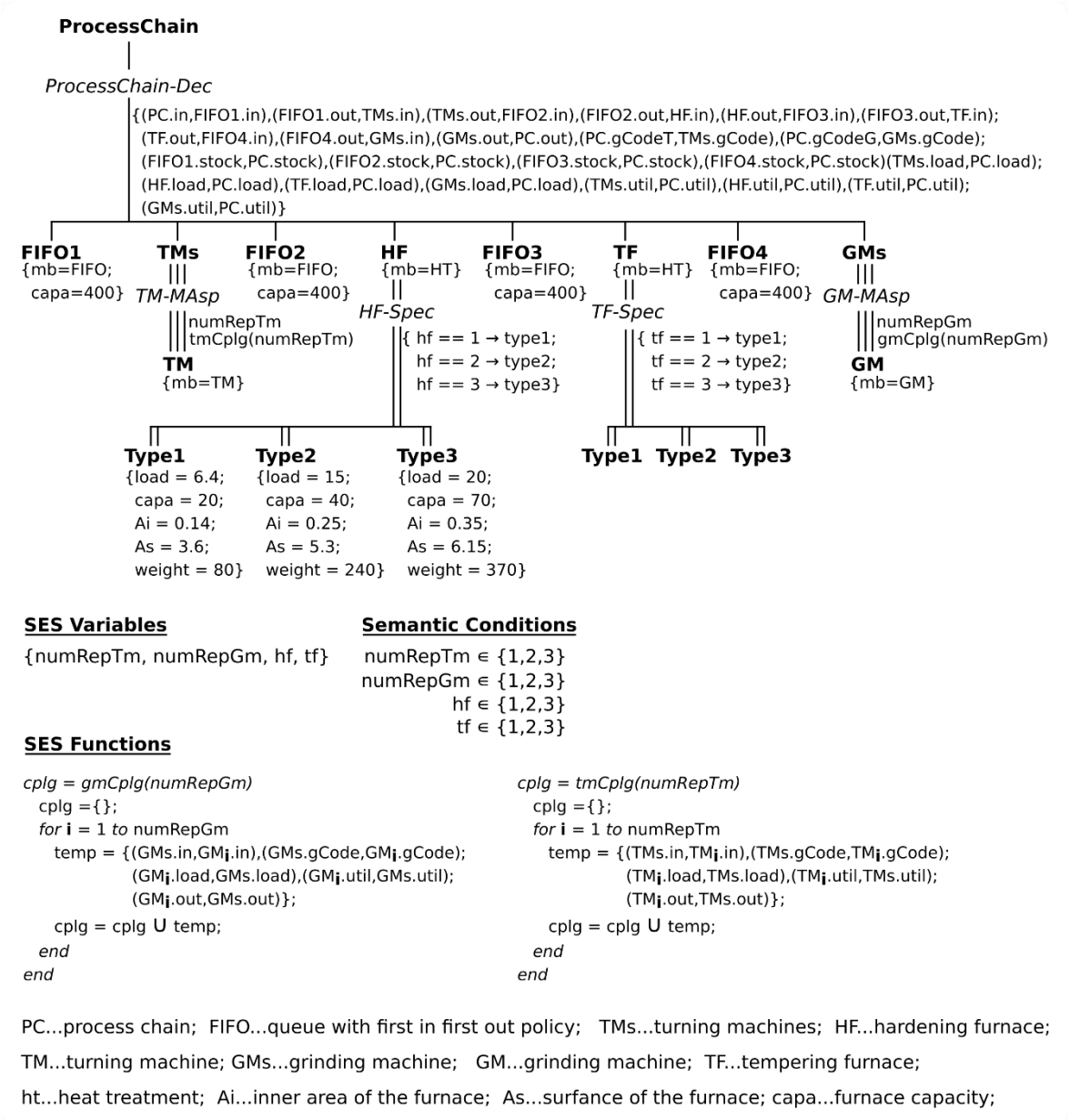


Abbildung 48: Formale Spezifikation der MUS Varianten mit einer SES.

Weiterhin spezifiziert die SES die Konfigurationsparameter für die MUS Komponenten. Der deskriptive Knoten *ProcessChain-Dec* beschreibt die Basiskomposition aller Prozessketten-strukturen, bestehend aus den Entitäten: *FIFO1*, *TMs*, *FIFO2*, *HF*, *FIFO3*, *TF*, *FIFO4* und

GMs, und definiert die Kopplungsbeziehungen. Die Entitäten-Knoten *TMs* und *GMs* spezifizieren mittels des jeweils nachfolgenden Multi-Aspekt-Knotens, *TM-MAsp* und *GM-MAsp*, ihre variable Komposition aus typgleichen *TM* sowie *GM* Komponenten. Dessen variable Anzahl wird mit den Attributen *numRepTm* sowie *numRepGm* und die daraus resultierenden variablen Kopplungsrelationen mit den SES-Funktionen *tmCplg(...)* sowie *gmCplg(...)* beschrieben. Die Entitäten-Knoten *HF* und *TF* definieren mit den nachfolgenden Spezialisierungsknoten, *HF-Spec* und *TF-Spec*, und deren Nachfolgern die drei spezifischen Ofentypen. Hierbei werden die unterschiedlichen Parametriermöglichkeiten der Ofentypen als Attribute der Entitäten *Type1*, *Type2* und *Type3* abgebildet. Die Attribute der Entitäten *Type1*, *Type2* und *Type3* im Teilbaum von *HF-Spec* gelten automatisch für die gleichnamigen Entitäten im Teilbaum von *TF-Spec*. Dieser Sachverhalt ist dem Axiom *Uniformität*, AX2 im Unterabschnitt 3.2.1, geschuldet. Die formale Kopplung zwischen Entitäten-Knoten in der SES und den Komponenten in der MB ist mit dem Attribut *mb* an den jeweiligen Entitäten-Knoten beschrieben. Die SES Variablen *numRepTm*, *numRepGm*, *hf* und *tf* bilden Eingangsvariablen der SES, die sogenannte Nutzerschnittstelle, welche durch die semantischen Bedingungen konkretisiert werden. In der Tabelle 11 sind die in der SES spezifizierten Variationspunkte auf Ebene des MUS nochmals zusammengefasst. Damit ist die Variantenanalyse und die konzeptuelle Modellierung von Prozesskettenvarianten auf MUS Ebene abgeschlossen.

Tabelle 11: Zusammenfassung der Variationspunkte der SES in Abbildung 48.

Variationspunkte		**Auswahlmöglichkeiten**	
Knoten	*SES-Funktionen*	*qualitativ*	*quantitativ*
TM-MAsp		TM1, (TM1, TM2), (TM1, TM2, TM3)	3
GM-MAsp		GM1, (TG1, TG2), (GM1, GM2, GM3)	3
HF-Spec		Type1, Type2, Type3	3
TF-Spec		Type1, Type2, Type3	3
	tmCplg(numRepTm)	vgl. Abbildung 48	3
	gmCplg(numRepGm)	vgl. Abbildung 48	3

6.2.2 Experimental Frame und Simulationsmodell

Zum Aufbau eines simulationsbasierten Experiments ist ein MUS mit einem Experimental Frame (EF) zu koppeln. Beide zusammen bilden das Simulationsmodell. Zum möglichst frühzeitigen Ausschluss nicht-zielführender Prozesskettenvarianten soll jede MUS Variante gemäß den drei Experimentphasen – frühzeitig, mittelfristig, langfristig – untersucht werden. In der ersten Phase soll ein Screening, in der zweiten Phase eine Sensitivitätsanalyse und in der dritten Phase eine Parameteroptimierung erfolgen. Hierzu sind die notwenigen Experimental Frames, mit den Komponenten Generator, Transducer und Acceptor zu entwickeln. Bei der Analyse zum Aufbau der EF zeigte sich, dass für das Screening und die Sensitivitätsanalyse ein identischer EF nutzbar ist. Abbildung 49 zeigt die Struktur der beiden EF mit integriertem MUS. Weiterhin ist in Abbildung 49 ersichtlich, dass beide EF

Strukturen eine identische Komponente Generator verwenden und kein separater Acceptor notwendig sind. Damit beschränkt sich die EF-Variabilität auf zwei unterschiedliche Transducer-Komponenten und differenzierte Kopplungsbeziehungen.

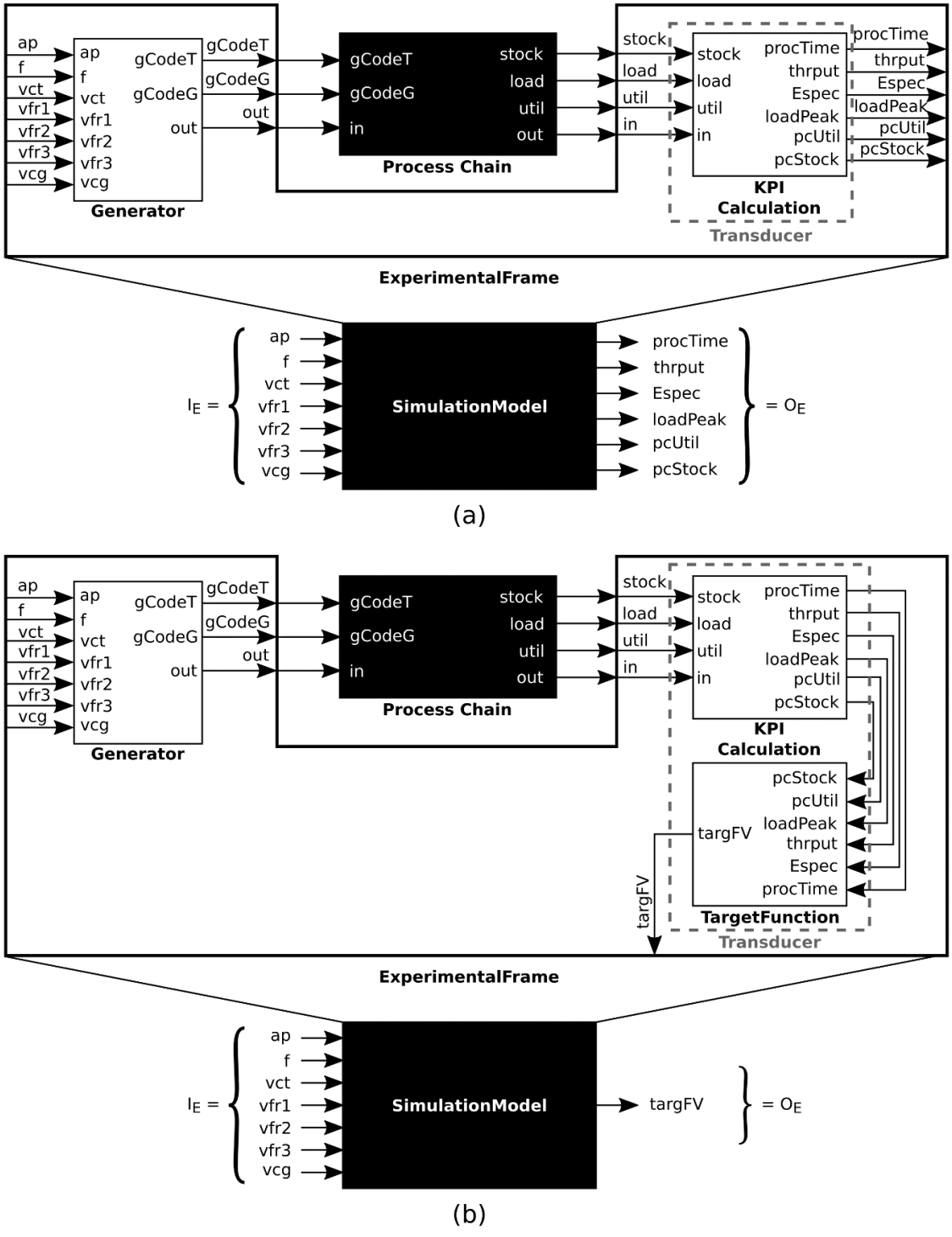


Abbildung 49: Schematische Darstellung der Experimental Frames und der resultierenden Simulationsmodelle für die Experimentziele: (a) Screening sowie Sensitivitätsanalyse und (b) Parameteroptimierung.

Der *Generator* erstellt auf Basis der Untersuchungsgrößen (*ap, f, vct, vfr1,vfr2,vfr3,vcg*), die für das MUS notwendigen G-Codes (*gCodeT*, *gCodeG*), und liefert die Rohlinge über die Kopplung *(Generator.out, ExperimentalFrame.out)*. Der Transducer *KPICalculation*

berechnet aus den Ausgabegrößen des MUS die geforderten Zielgrößen Durchlaufzeit (*procTime*), Durchsatz (*thrput*), Energie pro Fertigteil (*Espec*), elektrische Lastspitze (*loadPeak*), Auslastung (*pcUtil*) und Bestand (*pcStock*). Im Fall der Optimierung besteht der Transducer aus der *KPICalculation* und der Zielfunktionswertberechnung (*TargetFunction*). Die Transducer-Subkomponente *TargetFunction* bildet aus den einzelnen Zielgrößen den Zielfunktionswert gemäß Gleichung 22. Abbildung 50 zeigt die MB mit den drei EF-Komponenten *Generator, KPICalculation*, *TargetFunction*.

Auf der Ebene des Simulationsmodells unterscheiden sich die beiden Varianten (a) und (b) in Abbildung 49 nur in deren Ausgabegrößen. Die Eingabegrößen sind jeweils identisch und entsprechen den Untersuchungsgrößen (*ap, f, vct, vfr1, vfr2, vfr3, vcg*).

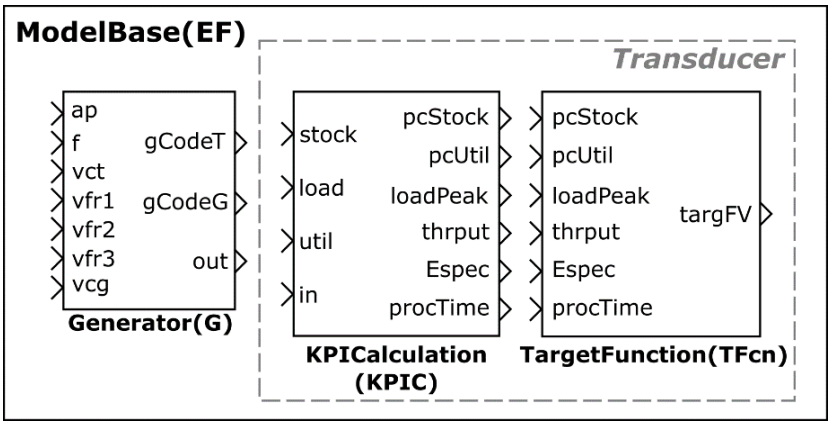


Abbildung 50: Modellbibliothek der EF-Komponenten.

Abbildung 51 zeigt die Spezifikation für alle Simulationsmodellvarianten mit einer SES. Demnach zerlegt sich die Entität *SimulationModel* in die Entitäten *ProcessChain*, das MUS, und *ExperimentalFrame*. Letztere wird in *Generator* und *Transducer* aufgeteilt. Die Entität *ProcessChain* ist ein Merge-Punkt zur SES in Abbildung 48, welche die MUS Varianten spezifiziert. Der strukturelle Aufbau des Transducers unterscheidet sich je nach Experimentphase und damit nach der verwendeten Experimentmethode. Diesen Sachverhalt kennzeichnet die Auswahlregel zwischen den Knoten *Transducer-Dec1* und *Transducer-Dec2*. Im Fall von Screening und Sensitivitätsanalyse besteht der Transducer nur aus der Entität *KPICalculation(KPIC)*. Im anderen Fall ergibt sich der Transducer aus den Entitäten *KPICalculation* sowie *TargetFunction.* Die Aspekt-Knoten *SimulationModel-Dec* und *ExperimentalFrame-Dec* benutzen in ihren Attributen jeweils eine SES-Funktion *cplgFunSimM(em)* sowie *cplgFunEF(em)* und spezifizieren damit die Variabilität der Kopplungsrelationen, die sich aus den unterschiedlichen Experimentphasen ergibt. Tabelle 12 fasst die Variationspunkte der SES zusammen. Damit ist die Analyse und formale Modellierung der Simulationsmodellvarianten abgeschlossen.

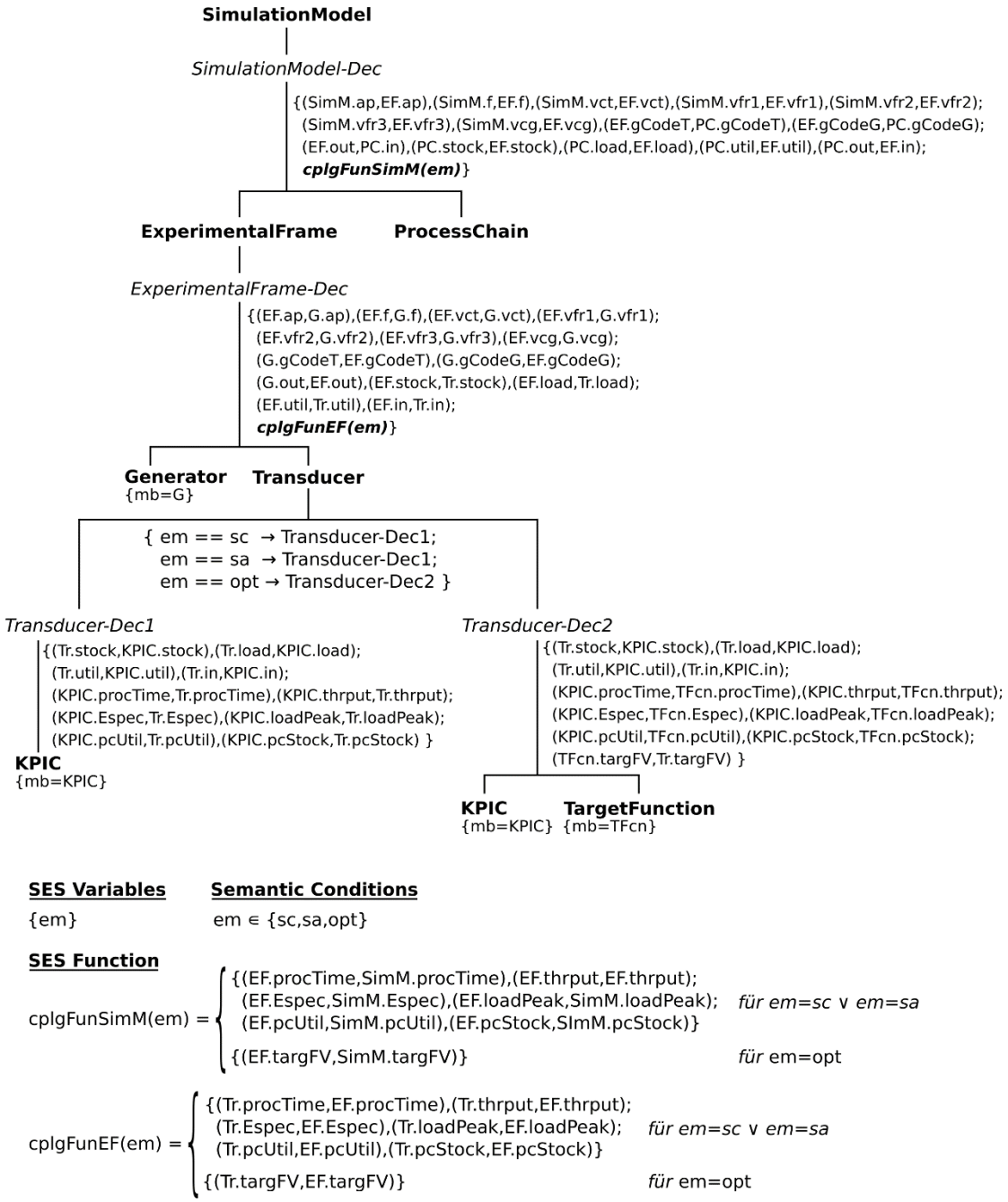


Abbildung 51: SES aller Simulationsmodellvarianten. Der Knoten *ProcessChain* ist ein Merge-Punkt zur SES der MUS Varianten in Abbildung 48.

Tabelle 12: Zusammenfassung der Variationspunkte der SES in Abbildung 51.

Variationspunkte		**Auswahlmöglichkeiten**	
Knoten	*SES-Funktionen*	*qualitativ*	*quantitativ*
Transducer		Transducer-Dec1, Transducer-Dec2	2
	cplgFunSimM(em)	vgl. Abbildung 51	2
	cplgFunEF(em)	vgl. Abbildung 51	2

6.2.3 Simulations- und Experimentmethoden

Die zu lösende Problemstellung gehört nach Kapitel zwei zur Klasse der *komplexen Experiemente*. Komplexe Experimente besitzen eine *Experimentmethode*, die unter Verwendung einer Simulationsmethode automatisiert Simulationsläufe ausführt, die ermittelten Simulationsergebnisse analysiert, speichert und gegebenenfalls weitere Simulationsläufe startet und auswertet. Aus der Problemspezifikation folgen zwei Varianten komplexer Experimente. Abbildung 52 zeigt die Struktur der Experimentvarianten: (a) Screening sowie Sensitivitätsanalyse und (b) Optimierung.

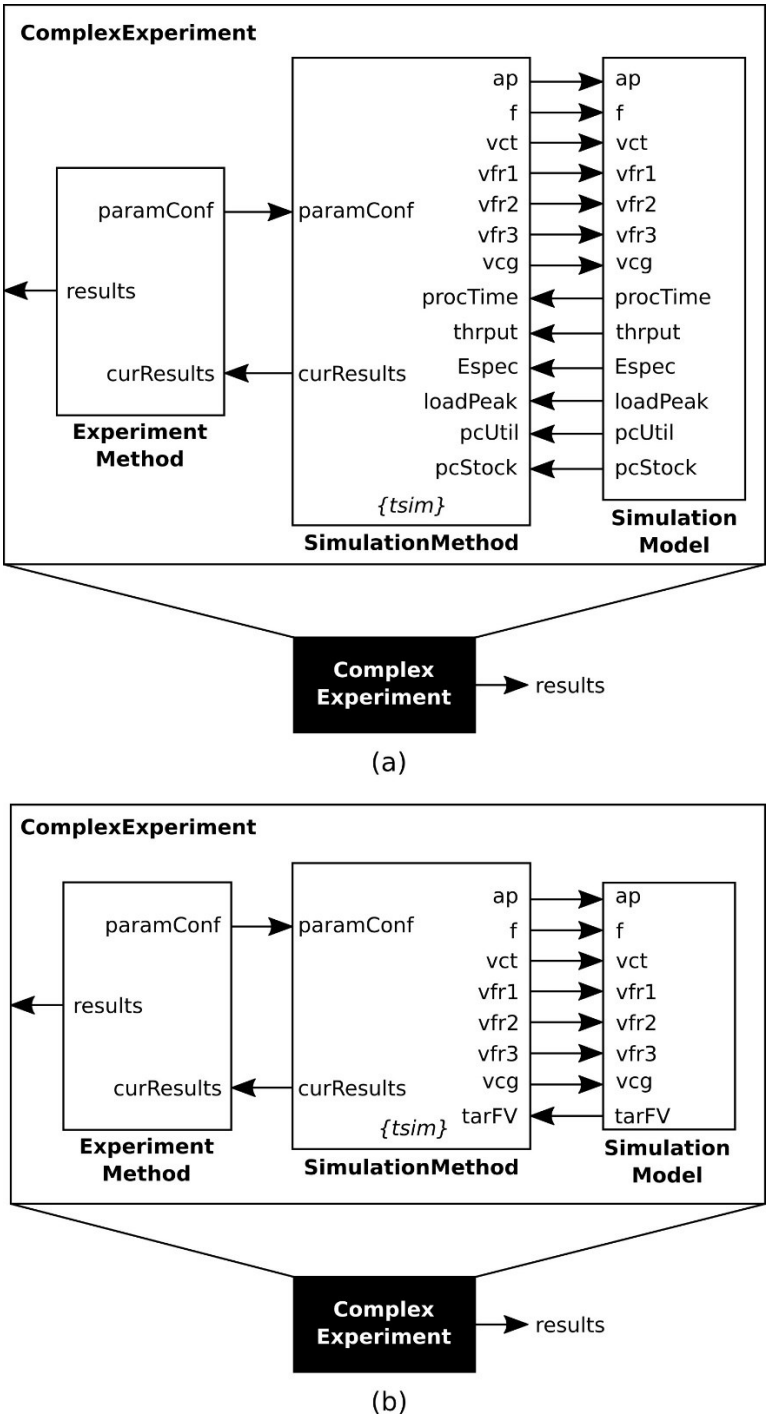
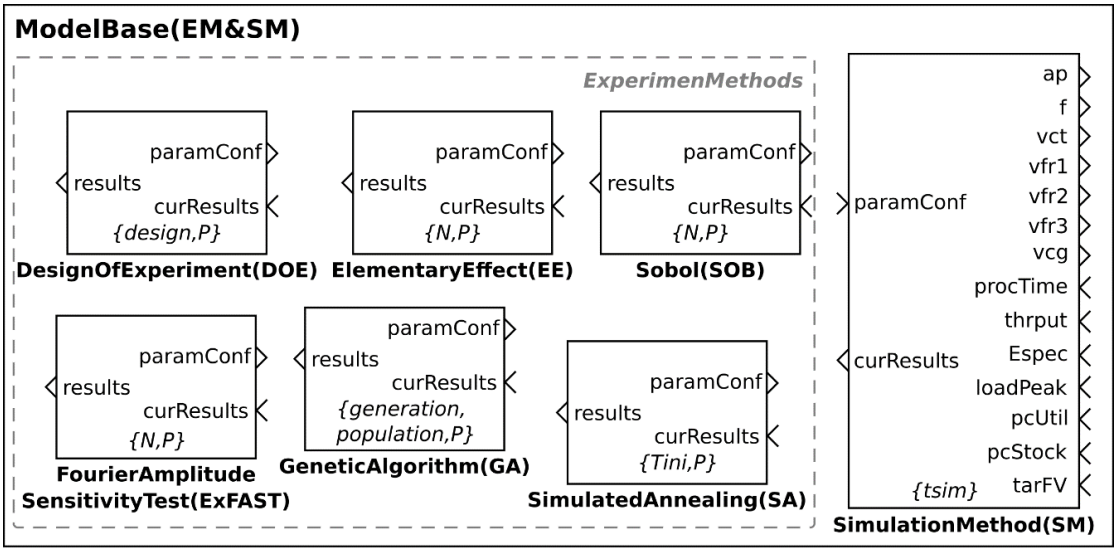


Abbildung 52: Schematische Darstellung der beiden komplexen Experimentstrukturen: (a) für das Screening sowie Sensitivitätsanalyse und (b) für eine Optimierung.

Neben unterschiedlichen Ausprägungen der Komponente *ExperimentMethod*, besteht kein weiterer Unterschied der beiden Experimentstrukturen in der Schnittstelle zwischen den Komponenten *SimulationMethod* und *SimulationModel*. Die unterschiedlichen Schnittstellen folgen aus den differenzierten Transducer-Komponenten im EF, wie im letzten Unterabschnitt diskutiert.

Abbildung 53 zeigt die Modellbibliothek mit den konkret ausgewählten und implementierten Komponenten *ExperimentMethod*, einschließlich einer generischen Komponente *SimulationMethod* für beide Experimentstrukturen in Abbildung 52. Die generische *SimulationMethod* Komponente implementieret alle notwendigen Schnittstellen. Je nach Experimentphase ergibt sich eine andere Verschaltung der Komponenten *SimulationModel* und *SimulationMethod*. Darüber hinaus besitzt die Komponente *SimulationMethod* einen methodenspezifischen Parameter *tsim*, welcher die Dauer eines Simulationslaufes und damit den Untersuchungszeitraum codiert. Nach der Problemspezifikation in Abschnitt 6.1 soll der Untersuchungszeitraum zwei Wochen betragen. Als *ExperimentMethod* wurden für das Screening die Methoden *Design of Experiment* sowie *Elementary Effect* gemäß [132], für die Sensitivitätsanalyse die *Sobol Methode* sowie *Fourier Amplitude Sentivity Test* nach [20] und für die Optimierung ein *Genetischer Algorithmus* sowie das *Simulated Annealing* Verfahren analog zu [155] ausgewählt.

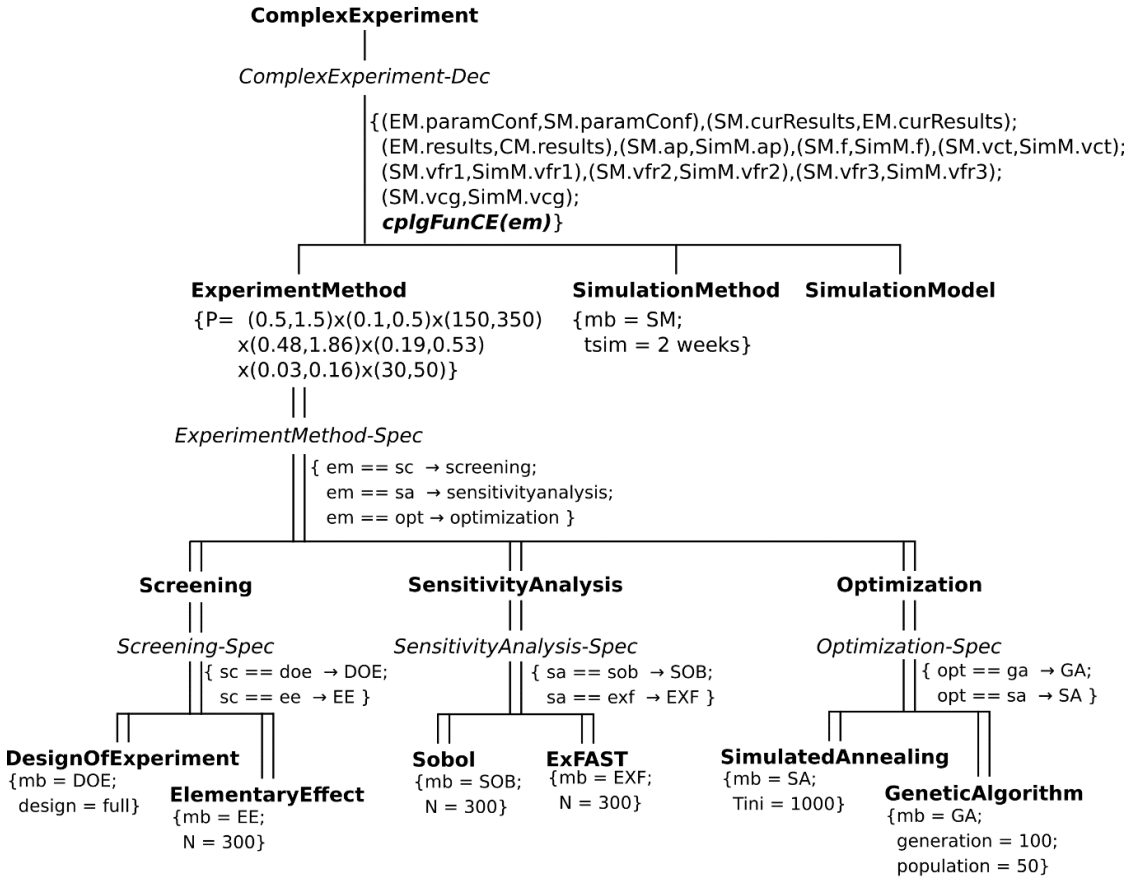


N...sample size; P...parameter space; Tini...initial temperature; tsim...simulation time

Abbildung 53: Modellbibliothek der ausgewählten Experimentmethoden einschließlich einer generischen Simulationsmethode.

Aus den sechs ausgewählten Experimentmethoden folgen 6 Experimentstrukturen, wobei die beiden kurzfristigen (Screening) sowie die zwei mittelfristigen (Sensitivitätsanalyse) dem Aufbau in Abbildung 52a folgen und die zwei langfristigen (Optimierung) dem Aufbau in Abbildung 52b. Die unterschiedlichen Strukturen der komplexen Experimente und deren Konfiguration sind mit der SES in Abbildung 54 spezifiziert. Demnach wird jedes komplexe Experiment, repräsentiert durch die Wurzel-Entität *ComplexExperiment*, in die Entitäten *ExperimentMethod*, *SimulationMethod* und *SimulationModel* zerlegt. Wobei die Entität *ExperimentMethod* den für die Untersuchung notwenigen Parameterraum *P* als Attribut spezifiziert. Die Zeitdauer jedes Simulationslaufes ist im Attribut *tsim* der Entität

SimulationMethod codiert und die Verlinkung mit der Komponente in der Modellbibliothek im *mb* Attribut. Die Zerlegung der Entität SimulationModel entspricht der SES in Abbildung 51. Die Entität *ExperimentMethod* wird mit dem Spezialisierungs-Knoten *ExperimentMethod-Spec* in die Entitäten *Screening*, *SensitivityAnalysis* und *Optimization* spezialisiert, welche jeweils weiter spezialisiert werden. Die Auswahlregeln an den Spezialisierungs-Knoten definieren die Methodenauswahl in Abhängigkeit von SES-Variablenbelegungen. Der Aspekt-Knoten *ComplexExperiment-Dec* nutzt eine SES-Funktion *cplgFunCE(em)*, welche die Variabilität bezüglich der möglichen Kopplungsrelationen in Abhängigkeit der ausgewählten Experimentmethode spezifiziert. Tabelle 13 fasst die Variationspunkte übersichtlich zusammen.



SES Variables

{em, sc, sa, opt}

Semantic Conditions

em ∈ {sc,sa,opt}
sc ∈ {doe,ee}
sa ∈ {sob,exf}
opt ∈ {ga,sa}

SES Function

$$\text{cplgFunCE(em)} = \begin{cases} \{(\text{SimM.procTime},\text{SM.procTime}),(\text{SimM.thrput},\text{SM.thrput}); \\ \quad (\text{SimM.Espec},\text{SM.Espec}),(\text{SimM.loadPeak},\text{SM.loadPeak}); \\ \quad (\text{SimM.pcUtil},\text{SM.pcUtil}),(\text{SimM.pcStock},\text{SM.pcStock})\ \} & \textit{für } em{=}sc \vee em{=}sa \\ \{(\text{SimM.targFV},\text{SM.targFV})\} & \textit{für } \text{em=opt} \end{cases}$$

CE...complex experiment; SM...simulation method; SimM...simulation model; tsim...simulation time;
EM...experiment method; SC...screening; SA...sensitivity analysis; OPT...optimization;
DOE...design of experiment; EE..elementary effect; GA...genetic Algorithm; SA...simulation annealing;
SOB...sobol; EXF...extendet fourier amplitude sensitivity test; N...sampling size; Tini...initial temperature

Abbildung 54: System Entity Structure aller Varianten komplexer Experimente mit Links zur Modellbibliothek in Abbildung 53.

Tabelle 13: Zusammenfassung der Variationspunkte der SES in Abbildung 54.

Variationspunkte		**Auswahlmöglichkeiten**	
Knoten	*SES-Funktionen*	*qualitativ*	*quantitativ*
EM-Spec		Screening, SensitivityAnalysis, Optimization	2
SC-Spec		DesignOfExperiment, ElementaryEffect	2
SA-Spec		Sobol, ExFAST	2
OPT-Spec		GeneticAlgorithm, SimulatedAnnealing	2
	cplgFunCE(em)	vgl. Abbildung 54	2

Im nächsten Unterabschnitt werden die Ergebnisse der Variantenanalyse sowie der konzeptuellen Modellierung zusammengefasst.

6.2.4 Zwischenstand

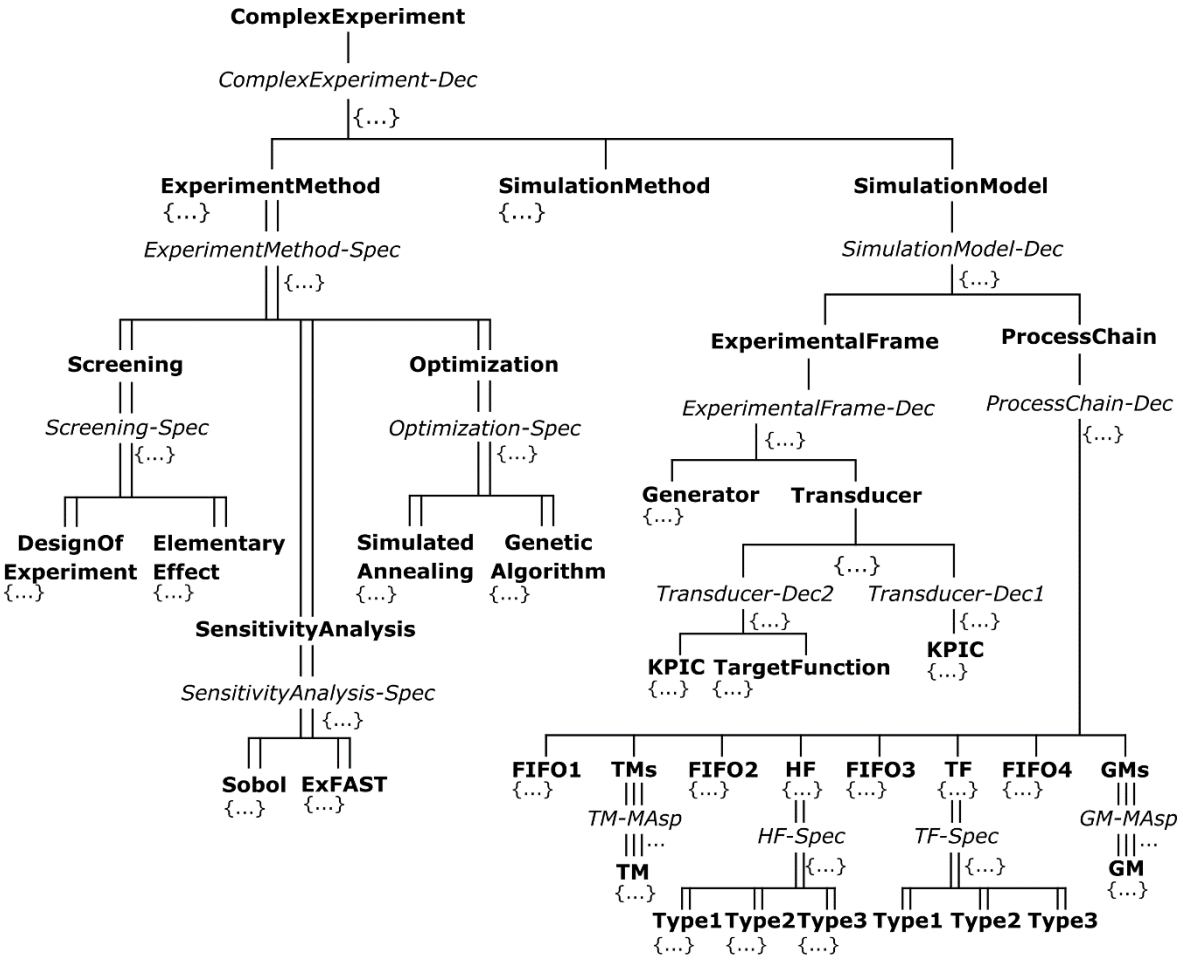
Mit Abschluss der Variantenanalyse liegt ein vollständiges konzeptuelles Modell aller Varianten komplexer Experimente in Form einer SES und der zugehörigen MB vor. Abbildung 55 zeigt schematisch die SES sowie die zugehörige MB. Die SES-Variablen bilden die Schnittstelle zum automatisierten Experimentieren unter Verwendung des in Kapitel 6 vorgestellten Konzeptrahmens. Die Wertebereiche der SES-Variablen werden durch die semantischen Bedingungen konkretisiert. Mit den 81 unterschiedlichen Prozesskettenausprägungen (Teilbaum *ProcessChain*) und den verschiedenen Experimentmethoden (Teilbaum *ExperimentMethod*) spezifiziert die SES insgesamt 486 Varianten.

In Analogie zu Gleichung 13 auf der Seite 72, ist der Parameterraum der SES-Variablen der Abbildung 55 in Gleichung 24 zusammengefasst.

$$P_{SESVar} = \{sc, sa, opt\} \times \{doe, ee\} \times \{sob, exf\} \times \{ga, sa\} \times \{1,2,3\} \times \ldots \\ \{1,2,3\} \times \{1,2,3\} \times \{1,2,3\} \quad (24)$$

Der Parameterraum bildet die Grundlage zur Variantengenerierung durch die Methoden *pruning* und *transaltion* gemäß Unterabschnitt 3.3.5 und wird von der Experimentsteuerung verwaltet.

Im nächsten Abschnitt wird die prinzipielle Implementierung der SES mit der zugehörigen MB in MATLAB/Simulink/SimEvents diskutiert. Zur Spezifikation der SES wird der im Rahmen dieser Arbeit entwickelte SES-Editor [117, 114] verwendet. Wie bereit am Anfang dieser Arbeit beschrieben, wird für die Erstellung der MB die Version MATLAB R2014b und damit die SimEvents-Bibliothek 4.3.3 verwendet.



SES Variables

{em, sc, sa, opt, numRepTm, numRepGm, hf, tf}

SES Function

cplgFunCE(em); cplgFunSimM(em); cplgFunEF(em);
tmCplg(numRepTm); gmCplg(numRepGm)

Semantic Conditions

em ϵ {sc,sa,opt}	numRepTm ϵ {1,2,3}
sc ϵ {doe,ee}	numRepGm ϵ {1,2,3}
sa ϵ {sob,exf}	hf ϵ {1,2,3}
opt ϵ {ga,sa}	tf ϵ {1,2,3}

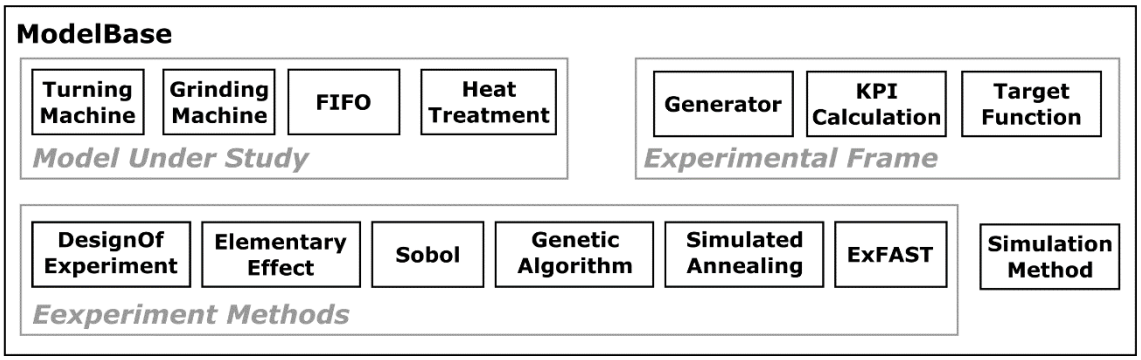


Abbildung 55: Schematische Darstellung der gesamten SES mit zugehöriger MB.

6.3 Umsetzung der System Entity Structure und der Modellbibliothek in MATLAB/Simulink/SimEvents

Die prinzipielle Umsetzung komplexer Experimente in MATLAB/Simulink ist in Abbildung 56 schematisch dargestellt. Demnach werden die Komponenten der Kategorie *Experiment*- und *SimulationMethod* in MATLAB mittels der *M-launguage* als Funktionen umgesetzt. Die Simulationsmodelle (Kategorie *SimulationModel*) werden als Simulink/SimEvents Modelle abgebildet. Der Grund für die Verwendung von M-Funktionen liegt darin, dass eine große Menge von Experimentmethoden für MATLAB in Form von M-Funktionen zur Verfügung steht. Beispiele dafür sind: (i) die *Global Optimization Toolbox* [159], (ii) die *Statistics and Machine Learning Toolbox* [161] oder die umfangreichen Implementierungen unter *MATLABCenter-FileExchange* [160].

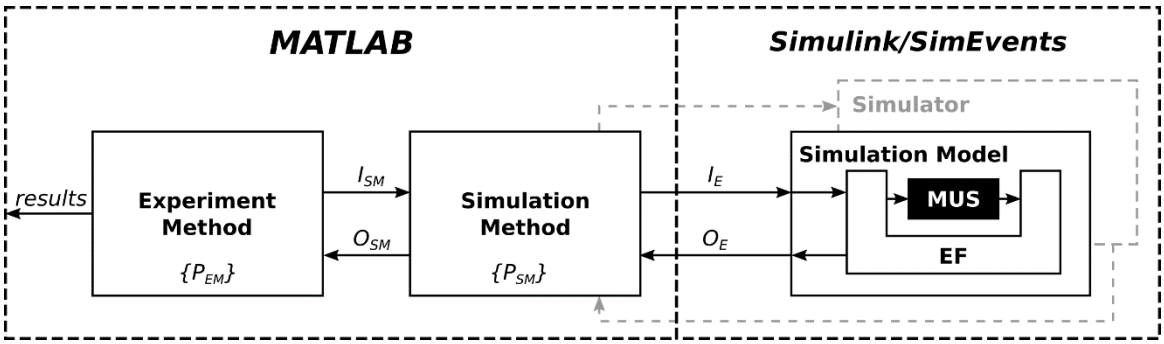


Abbildung 56: Prinzipielle Umsetzung der komplexen Experimente in MATLAB/Simulink.

Abbildung 57 zeigt die in MATLAB implementierte Modellbibliothek mit: (i) den Komponenten für die Experimentmethoden (*.m Dateien) sowie (ii) der generischen Simulationsmethode (*.m Datei) und (iii) den Komponenten für die Simulationsmodelle der Prozessketten mit Experimental Frame (*simModelBase.slx* Archivdatei).

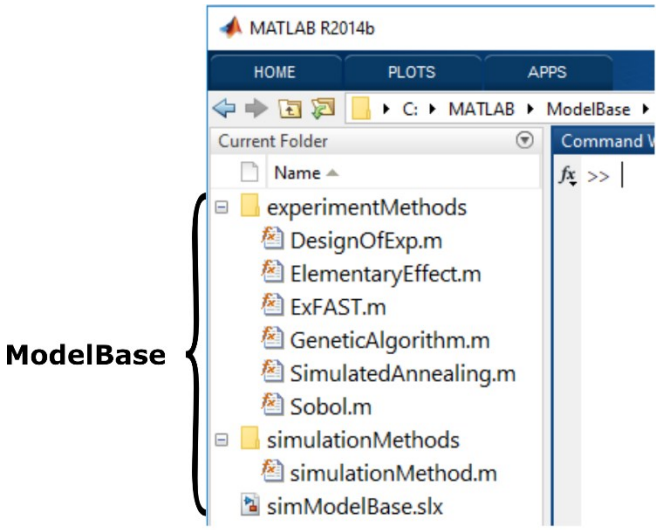


Abbildung 57: Umsetzung der Modellbibliothek aus Abbildung 55 in MATLAB.

Alle Experimentmethoden, außer der *Sobol Methode*, sind Implementierungen Dritter. Die Experimentmethode *DesignOfExperiment* basiert auf der *Statistics and Machine Learning Toolbox* [161], *ElementaryEffect* entspricht der Implementierung nach [163], *ExFAST* ist

analog zu [36] implementiert und die *GeneticAlgorithm* sowie *SimulatedAnnealing* sind Methoden der *Global Optimization Toolbox* [159]. Die Experimentmethode *Sobol* wurde basierend auf der Theorie von Saltelli et al. [132] umgesetzt.

Abbildung 58 zeigt die implementierte Modellbibliothek *simModelBase.slx* mit Basiskomponenten zur Komposition unterschiedlicher Simulationsmodellvarianten.

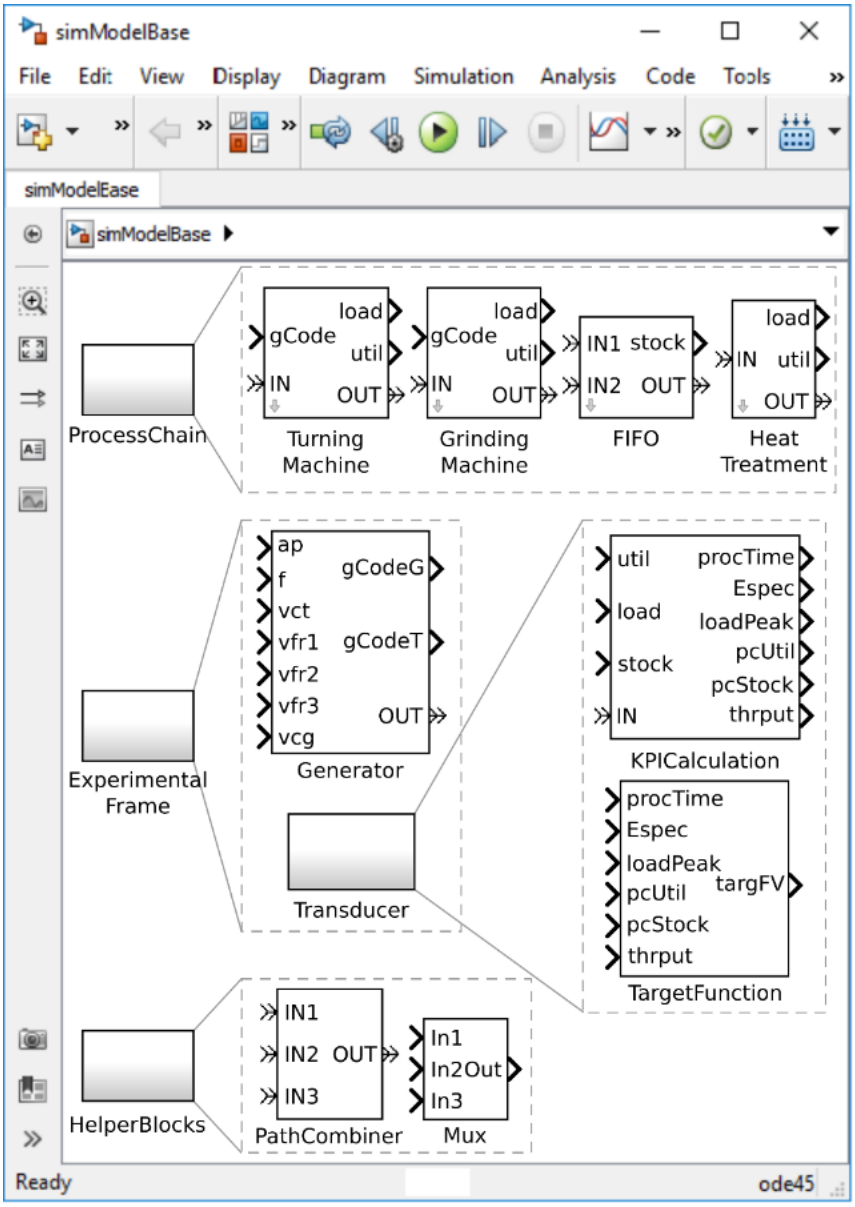


Abbildung 58: Simulink/SimEvents-basierte Modellbibliothek mit Basiskomponenten zur Generierung unterschiedlicher Simulationsmodelle für Prozesskettenvarianten (MUS) mit Experimental Frame (EF) gemäß Abschnitt 6.2.

Die Komponenten *TurningMachine* und *GrindingMachine* basieren auf ursprünglichen Arbeiten von Larek [87]. Sie wurden im Rahmen der Arbeit zur Senkung der Simulationslaufzeiten und zur Behebung von *Race Condition* Problemen grundlegend überarbeitet, wie in [138] publiziert. Die Komponente *HeatTreatment* wurde neu entwickelt und ist analog den Ausführungen in Schmidt und Pawletta [142] implementiert. Neben den Komponenten für das MUS (*ProcessChain*) und das Experimental Frame enthält die Modellbibliothek in

Abbildung 58 noch zwei Hilfsblöcke (*HelperBlocks*). Die Notwendigkeit der Hilfsblöcke ist in Abbildung 59 gezeigt. Anders als in der Abbildung 46 angenommen, können in MATLAB/Simulink nicht mehrere Eingangskopplungen an einem Eingangsport erfolgen. Für eine solche Realisierung ist beispielsweise ein *Multiplexer* (Mux) oder ein *Path Combiner* Block notwendig. Somit sind die Hilfsblöcke notwendige spezifische Komponenten für die MATLAB/Simulink Umgebung.

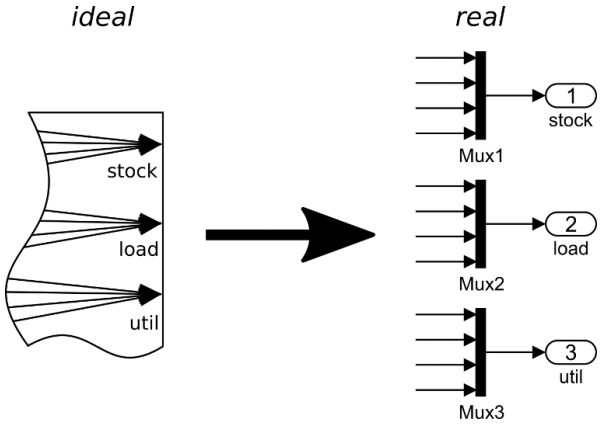


Abbildung 59: Darstellung der idealen Kopplungen gemäß Abbildung **46** und deren Umsetzung in MATLAB/Simulink.

Abschließend zeigt die Abbildung 60 ausschnittsweise die Umsetzung der SES aus Abbildung 55, welche alle Experimentvarianten spezifiziert. Links ist der gesamte SES-Baum dargestellt, erweitert um die MATLAB/Simulink spezifischen Entitäten *PathCombiner*, *MuxStock*, *MuxLoad*, *MuxUtilization*. Rechts sind die für das Experimentieren notwendigen SES-Variablen und die zugehörigen semantischen Bedingungen gezeigt. Darüber hinaus sind rechts: (i) die Umsetzung der Kopplungsrelationen am Aspekt-Knoten *T1Dec*, (ii) die Auswahlregeln am HFSpec Knoten, (iii) das charakteristische Attribut *mb* an der Entität *HardeningFurnace* und (iv) die Attributierung der Entität *Type1* dargestellt.

Im Anhang F ist die Umsetzung der Experimentsteuerung, gemäß Unterabschnitt 5.1.1, am Beispiel der frühzeitigen Experimentphase mit den Experimentmethoden für das Screening dargestellt. Nachfolgend erfolgt eine Diskussion der Ergebnisse.

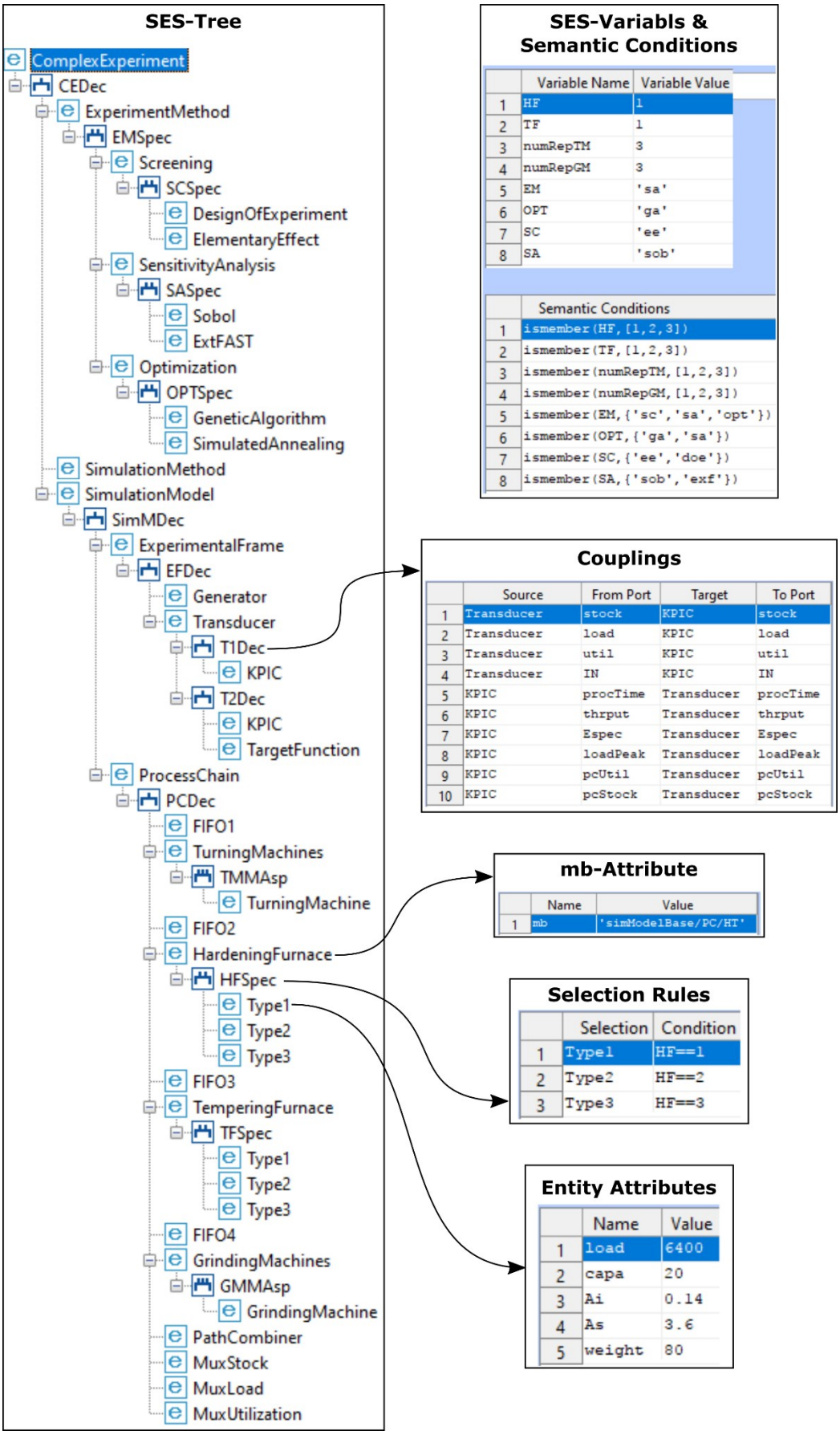


Abbildung 60: Ausschnitt der Implementierung der SES für alle Experimentvarianten unter Verwendung des SES-Editors für MATLAB/Simulink nach [117, 114].

6.4 Diskussion der Ergebnisse

Mit der vorgestellten Implementierung in MATLAB/Simulink wurden Experimente entsprechend den Experimentzielen durchgeführt. Die erzielten Ergebnisse, aufgetretene Probleme und Lösungsvorschläge werden nachfolgend dargestellt.

6.4.1 Experimentziel Screening

Die Ergebnisse für die kurzfristige Experimentphase mit der Experimentmethode *Design of Experiment* zeigt die Abbildung 61. Es wurden alle 81 Prozesskettenvarianten hinsichtlich der im Abschnitt 6.1 definierten Zielgrößen untersucht. Wie im Unterabschnitt 2.2.1 diskutiert, erfolgt beim Screening eine qualitative Berechnung der Sensitivitätsmaße. Damit erfolgt eine Einteilung der Einflussgrößen nur in signifikant und nicht-signifikant. Für jede Zielgröße codiert die Abszisse die Experimentnummer und damit die Prozesskettenvariante. Auf der Ordinate sind die Untersuchungsgrößen, im Rahmen des Screenings *Faktor* genannt, gemäß Abschnitt 6.1 aufgeführt. Jede mit einem Quadrat markierte Stelle weist eine Signifikanz bezogen auf die jeweilige Zielgröße auf. Ist eine Stelle nicht besetzt, so ist dieser Faktor bezogen auf die jeweilige Zielgröße nicht-signifikant. Beispielsweise werden alle Faktoren bei der Experimentnummern 45 bezüglich der Zielgröße *Bestand* als nicht-signifikant eingestuft. Zeitgleich aber weisen die gleichen Faktoren, bis auf den Faktor *vcg*, eine Signifikanz bezüglich der Zielgröße *Auslastung* auf. Um einen Faktor auf einen beliebigen Wert des zugehörigen Wertebereiches zu fixieren und ihn damit aus dem Parameterraum zu entfernen, muss dieser Faktor für alle Zielgrößen als nicht-signifikant eingestuft werden. Dieser Sachverhalt ist in der Abbildung 61 mit einem Kreuz markiert. Für die Experimentnummer 45 und 46 ist der Faktor *vcg* für alle Zielgrößen als nicht-signifikant eingestuft. Damit kann dieser Faktor bei den zugehörigen Prozesskettenvarianten auf einen beliebigen Wert des zugehörigen Wertebereiches fixiert werden und braucht in nachfolgenden Experimentphasen bei diesen beiden Prozesskettenvarianten nicht mehr betrachtet werden.

Leider konnten nur die 81 komplexen Experimente mit der Experimentmethode *Design of Experiment* praktisch durchgeführt werden. Somit liegen keine weiteren Berechnungsergebnisse vor. Die Gründe dafür werden im nächsten Unterabschnitt diskutiert.

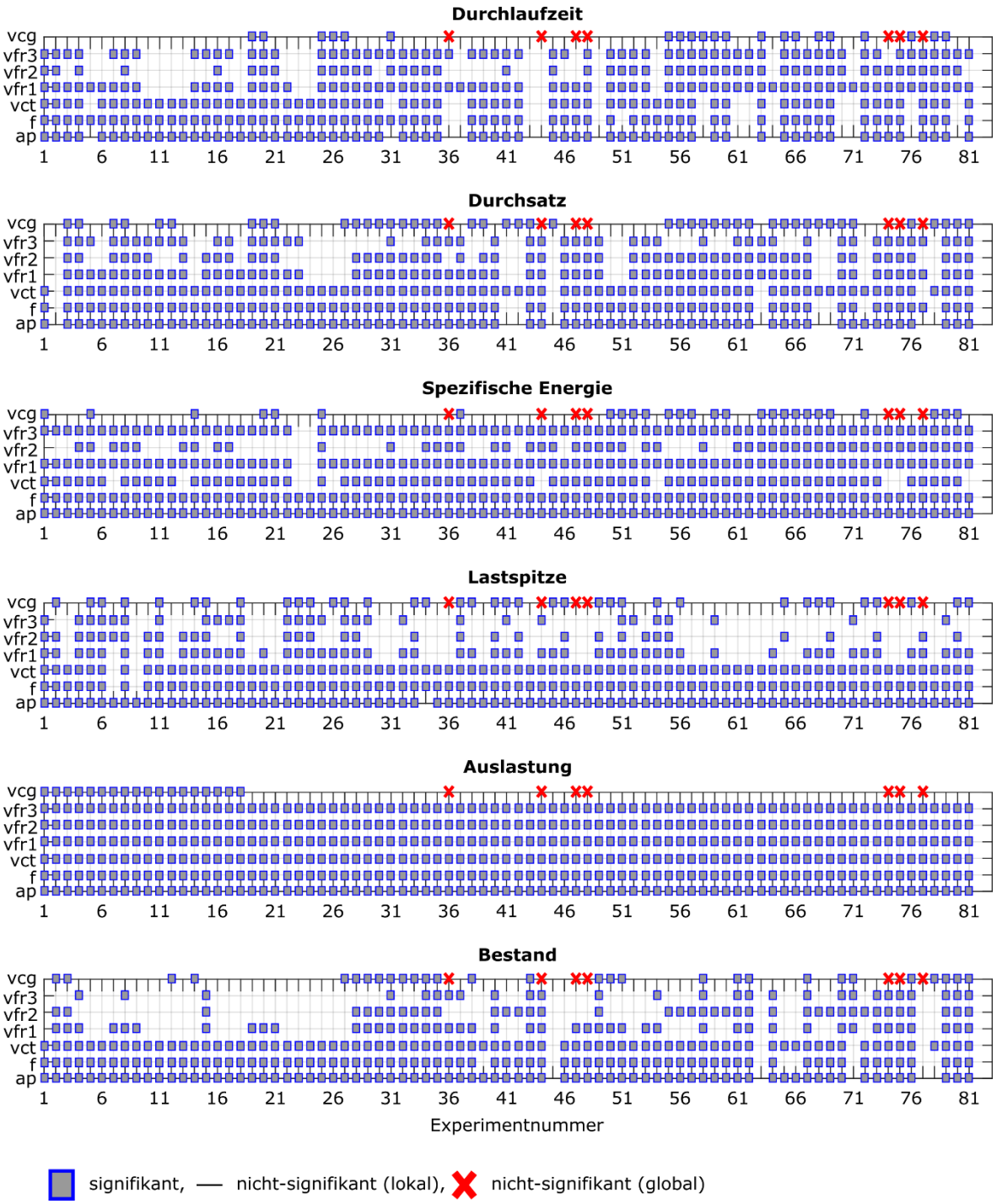


Abbildung 61: Einteilung der Faktoren bezüglich deren Signifikanz hinsichtlich aller Zielgrößen gemäß Abschnitt 6.1.

6.4.2 Probleme bei der Durchführung komplexer Experimente

Das Hauptproblem liegt in der hohen Zeitdauer eines Simulationslaufes (t_{SR}). Die empirisch ermittelte Zeitdauer auf einem gegenwärtigen Standard-PC für alle 81 Prozessketten-varianten ist in Gleichung 25 zusammengefasst.

$$t_{SR} \in [1{,}45; 5{,}4]\ [min] \tag{25}$$

Demnach beträgt die minimale Zeitdauer für einen Simulationslauf 1,45 und die maximale 5,4 Minuten. Dieser Sachverhalt erklärt sich mit dem sehr hohen Detaillierungsgrad der MUS-Komponenten *TurningMachine*, *GrindingMachine* und *HeatTreatment* in Abbildung 58. Der hohe Detaillierungsgrad wurde gewählt, um nach dem Vorbild von Larek [87] den Energieverbrauch möglichst spezifisch und detailliert bestimmen zu können. Bei den ersten beiden Komponenten werden jede einzelne Achsbewegung und damit jede einzelne Werkzeugbewegung als ein Ereignis modelliert. Für das Beispiel in Abbildung 44 und einer gewählten Schnitttiefe von 1 mm ergibt sich ein G-Code mit 62 Werkzeugbewegungen pro Bauteil und damit 62 Ereignissen pro Bauteil. Ähnliches gilt auch für die Komponente *GrindingMachine*. Bei der Komponente *HeatTreatment* ist neben dem Aufheizprozess auch der Abkühlprozess des Ofens detailliert modelliert, was sich auf die Simulationslaufzeit auswirkt, zumal jeweils zwei Öfen, einer für das Volumenhärten und einer für das Spannungsarmglühen, in einer Prozesskette eingesetzt werden. Im Anhang G wird eine Überschlagsrechnung für die Experimentausführungszeit (t_E) für die komplexen Experimente mit den Experimentmethoden *Elementary Effect* und *Sobol* für die 81 Prozesskettenvarianten aufgeführt. Deren Ergebnis ist in Gleichung 26 gezeigt.

$$t_E \in [70{,}4; 262]\ [Tage] \tag{26}$$

Demnach beträgt die minimale Experimentausführungszeit 70,4 und die maximale 262 Tage. Wobei der zeitliche Rechenaufwandt der Experimentmethoden *Simulatated Annealing* und *Genetic Algorithm* hierbei vernachläßigt ist.

Da im Rahmen der Arbeit keine passende Parallelrechentechnik zur Verfügung stand, wurde versucht, eine suboptimale Lösung an Hand einer einfachen Datenanalyse für die vorliegende Aufgabenstellung zu finden.

6.4.3 Vorschlag für eine suboptimale Lösung

Bei der Durchführung der Screening-Experimente wurden für jeden Simulationslauf und jede Zielgröße Simulationsergebnisse aufgenommen. Diese sind für alle 81 Prozesskettenvarianten in der Abbildung 62 dargestellt. Anhand dieser Daten wird ein Vorschlag für eine suboptimale Lösung der Problemstellung mittels einer einfachen Datenanalyse erarbeitet.

Für die Auswertung der in Abschnitt 6.1 vorgestellten Zielfunktion müssen die einzelnen Zielgrößen normiert werden, um sie vergleichbar zu machen. Hierzu sind die einzelnen Normierungskonstanten (min-/max-Werte) zu identifizieren. Tabelle 14 fast die aus Abbildung 62 identifizierten Normierungskonstanten zusammen.

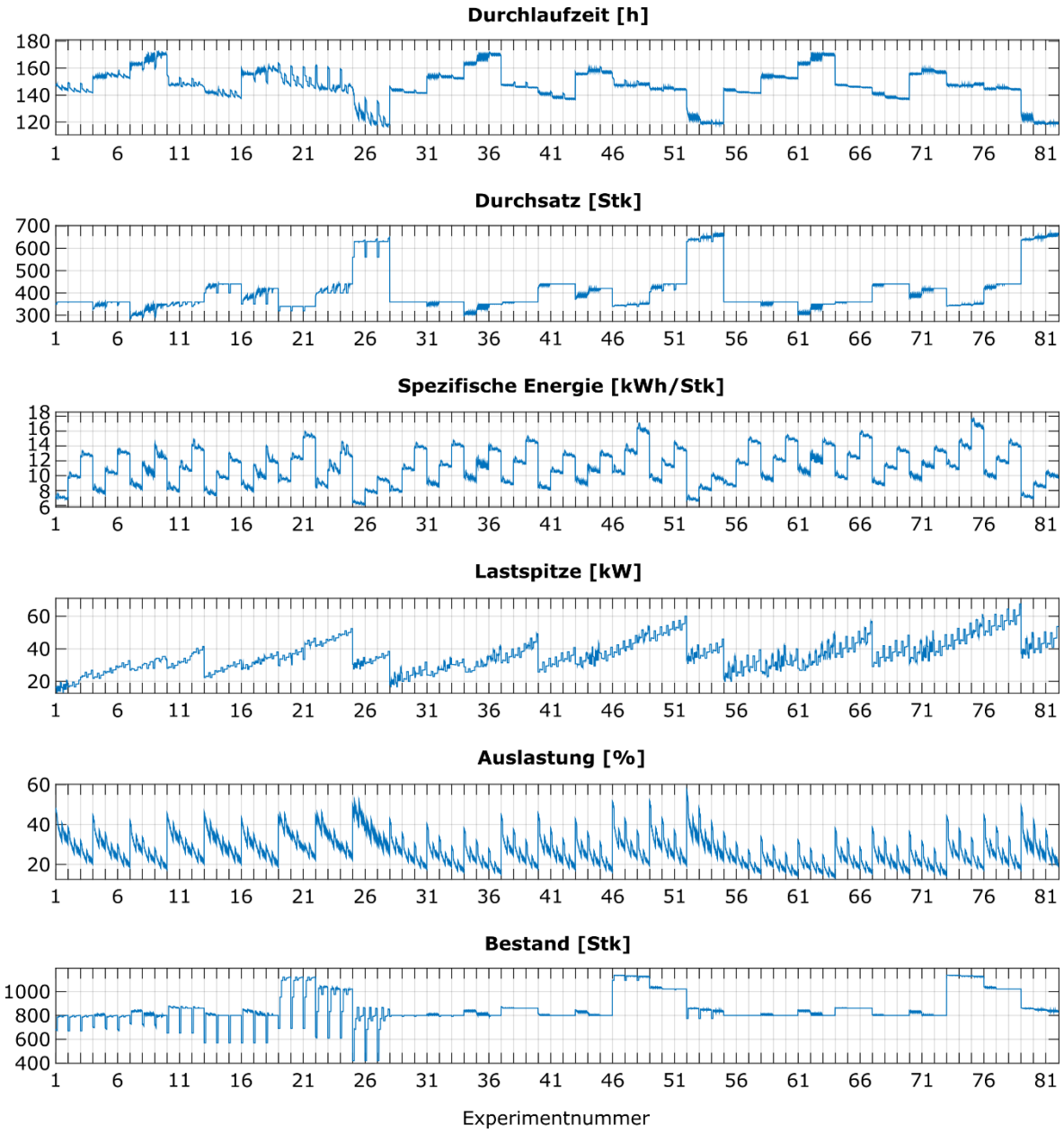


Abbildung 62: Detaillierte Übersicht der Zielgrößenverläufe für alle 81 Prozessketten-varianten.

Tabelle 14: Identifizierte Normierungskonstanten.

	Durchlaufzeit	**Durchsatz**	**Spezifische Energie**	**Lastspitze**	**Auslastung**	**Bestand**
Minimum	116.4	286	6.1	13.4	13.2	419
Maximum	172.4	669	17.7	67.7	57.3	1139

Nach der Normierung der einzelnen Zielgrößen kann die Zielfunktion gemäß Gleichung 22 berechnet werden. Abbildung 63 zeigt den Verlauf der Zielfunktion über alle Prozessketten-varianten (Experimentnummer).

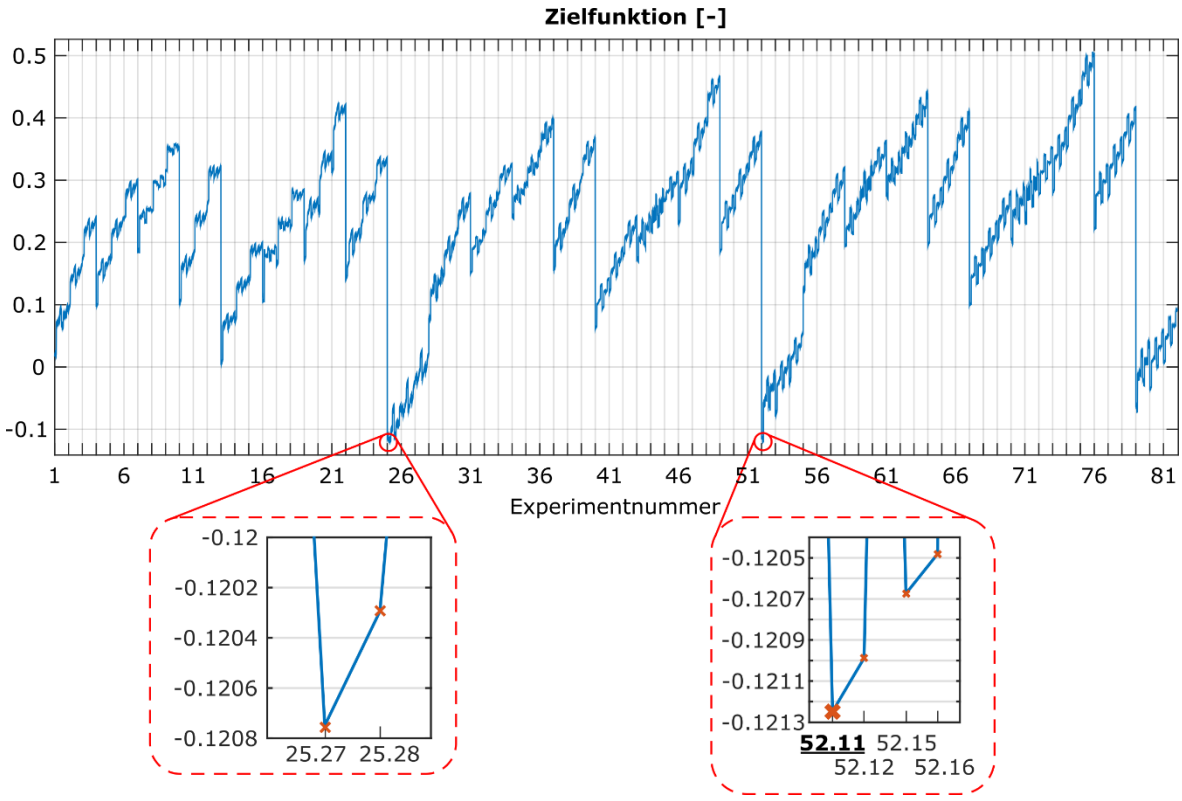


Abbildung 63: Verlauf der Zielfunktion über alle Prozesskettenvarianten.

Das globale Minimum liegt bei Experimentnummer 52 und den Simulationsergebnissen 11, kurz *52.11*. Dabei entspricht die Experimentnummer einer Prozesskettenvariante, exakt einer Prozesskettenstruktur, und die Nummer der Simulationsergebnisse einer spezifischen Konfiguration der Einflussgrößen. Weiterhin werden fünf lokale Minima *25.27*, *25.28*, *52.15*, *52.12*, *52.16*, die in der Nähe des globalen Optimums liegen, näher betrachtet und in der Tabelle 15 detailliert zusammengefasst. Wie bereits in Abbildung 63 zu sehen ist, liegen die Zielgrößenwerte sehr nahe bei einander. Bei näherer Betrachtung fällt jedoch auf, dass der wesentliche Unterschied bei der Anzahl der eingesetzten Drehmaschinen (Zeile TM) liegt. Die Ergebnisse der Experimentnummern *25.27* und *25.28* stechen hierbei besonders hervor. Denn diese benötigen eine Drehmaschine weniger als die anderen Varianten, jedoch ohne wesentliche Einbußen bei den Zielgrößen. Somit ist das Ergebnis der Variante *52.11*, obwohl es das globale Minimum darstellt, zugunsten der Variante *25.27* oder *25.28* zu verwerfen.

Tabelle 15: Detaillierte Zusammenfassung der ausgewählten Ergebnisse (* globales Minimum).

Experiment		52.11*	52.12	25.27	52.15	52.16	25.28
Zielfunktions-wert		-0.1213	-0.121	-0.1208	-0.1207	-0.1205	-0.1203
Struktur	TM	2	2	1	2	2	1
	HF	3	3	3	3	3	3
	TF	3	3	3	3	3	3
	GM	1	1	1	1	1	1
Untersuchungs-größen	ap	0.5	0.5	0.5	0.5	0.5	0.5
	f	0.1	0.1	0.1	0.1	0.1	0.1
	vct	150	150	350	150	150	350
	vfr1	1.86	1.86	1.86	1.86	1.86	1.86
	vfr2	0.19	0.19	0.19	0.53	0.53	0.19
	vfr3	0.16	0.16	0.16	0.16	0.16	0.16
	vcg	30	50	30	30	50	50
Zielgrößen	Durchlauf-zeit	126.3	126	130.4	125.5	125.3	130.2
	Durchsatz	630	630	630	630	630	630
	Spez. Energie	6.8	6.8	6.5	6.8	6.8	6.5
	Lastspitze	31	31	28.5	31	31	28.5
	Auslastung	52.3	52.1	46.5	51.5	51.3	46.2
	Bestand	774	774	682	774	774	682

6.5 Zusammenfassung

In diesem Kapitel wurde die entwickelte Methodik zum Variantenmanagement in der Modellbildung und Simulation auf Basis eines erweiterten SES/MB Frameworks an einem Beispiel aus dem Anwendungsbereich der Produktion und Logistik vorgestellt. Das übergeordnete Ziel war die Identifikation einer optimalen Prozesskettenvariante. Dazu wurden insgesamt 486 Varianten vom Typ komplexes Experiment, unter Beachtung der drei Experimentphasen: (i) frühzeitig, (ii) mittelfristig und (iii) langfristig, in einer SES modelliert sowie die notwendigen Komponenten implementiert und in einer MB organisiert. Gemäß der Spezifikation umfasste jede Experimentphase die Untersuchung von 81 unterschiedlichen Prozesskettenvarianten mit dem zugehörigen Parameterraum und mit zwei unterschiedlichen Experimentmethoden.

Beim Experimentieren mit einer hohen Anzahl an Varianten komplexer Experimente müssen eine erhebliche Anzahl an Simulationsläufen durchgeführt werden. Für die Problemstellung hat sich gezeigt, dass auf der verfügbaren Standardhardware die Ausführungszeit eines Simulationslaufes länger als fünf Minuten betragen kann und damit die Gesamtausführungszeit aller Varianten enorm hoch ist. Der Grund für die lange Ausführungszeit liegt in dem hohen Detaillierungsgrad der einzelnen Prozessketten-komponenten der Simulationsmodelle. Zur Lösung des Problems könnte: (i) der Detaillierungsgrad der Prozesskettenkomponenten reduziert werden oder (ii) eine Ausführung der Experimente auf Parallelrechentechnik erfolgen. Der erste Ansatz wurde bewusst nicht verfolgt. Im Rahmen des DFG geförderten Forschungsprojektes waren neue Ansätze zur detaillierten Bewertung des Energieverbrauchs von Prozesskettenmodellen zu entwickeln, besser gesagt es wurden bewusst Ansätze für einen angestrebten Detaillierungsgrad entwickelt. Die dazu prototypisch implementierten Prozesskomponenten wurden im Rahmen des Projektes in mehreren Phasen bezüglich ihres Laufzeitverhaltens optimiert und entsprechend reimplementiert. Eine weitere Laufzeitoptimierung der Komponenten wäre nur mit einer höheren Modellungenauigkeit erzielbar. Der zweite Ansatz ist dahingegen ohne Genauigkeitseinbußen umsetzbar. Die einzelnen komplexen Experimente sind größtenteils unabhängig voneinander parallel ausführbar. Auf einem verfügbaren Multicore-PC mit 8 Cores konnte gezeigt werden, dass sich durch die Parallelisierung komplexer Experimente nahezu immer ein linearer Speedup erreichen lässt, wenn auch ausreichend viel Hauptspeicher zur Verfügung steht.

Mit der verfügbaren Technik konnte nur das Experimentziel Screening mit der Experimentmethode *Design of Experiment* der frühzeitigen Experimentphase vollständig ausgeführt werden. Die damit gewonnenen Simulationsergebnisse wurden anschließend einer gewöhnlichen Datenanalyse unterzogen und es konnten dadurch bereits suboptimale Lösungen abgeleitet werden. Dieses Ergebnis unterstreicht die Sinnhaftigkeit des Experimentierens gemäß der drei Phasen und zeigt, dass nicht in jedem Fall enorm zeitaufwendige Optimierungsexperimente ausgeführt werden müssen.

Der theoretische Ansatz der entwickelten Methodik konnte vollständig anhand des Anwendungsbeispiels aufgezeigt werden. Eine vollständige praktische Umsetzung aller spezifizierten Experimentvarianten konnte aufgrund nicht verfügbarer Parallelrechentechnik nicht erbracht werden. Mit der stetigen Entwicklung der Rechentechnik werden derartige Problemstellungen voraussichtlich in naher Zukunft auf dann verfügbarer Standardrechentechnik lösbar sein.

7 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, eine Methode für ein Variantenmanagement in der M&S bis zur Ebene simulationsbasierter Experimente zu entwickeln. Dazu wurde zunächst der Status Quo identifiziert und eine Klassifikation simulationsbasierter Experimente eingeführt. Als Grundlage für die zu entwickelnde Methode wurde das in der M&S bereits bekannte, jedoch bisher nicht zum Variantenmanagement eingesetzte *System Entity Structure/Model Base Framework* (SES/MB) ausgewählt. Im klassischen Ansatz ist die MB eine Modellbibliothek, welche eine Menge konfigurierbarer Modellkomponenten mit definierten Ein- und Ausgangsschnittstellen wiederverwendbar organisiert. Die SES ist ein gerichteter azyklischer Graph, welcher unterschiedliche Modellstrukturen und Parametrierungen mit Links zu den Modellkomponenten in der MB, allgemein als formale Kopplung bezeichnet, spezifiziert. Weiterhin definiert das SES/MB Framework die abstrakten Methoden *pruning* und *translation* zur automatischen Generierung ausführbarer Simulationsmodelle.

Die Analyse des SES/MB Frameworks zeigte verschiedene offene Probleme, insbesondere hinsichtlich der formalen Kopplung zwischen SES und MB. Anhand eines konkreten Beispiels wurde gezeigt, dass nicht in jedem Fall eine eindeutige, ausführbare Variante eines Simulationsmodells generiert werden kann. Zur Überwindung der identifizierten Probleme wurden verschiedene Erweiterungen und Konkretisierungen eingeführt. Weiterhin wurden konkrete Algorithmen für die Methoden *pruning* und *translation* entwickelt. Darauf aufbauend wurde ein Softwareprototyp, die *SES-Toolbox for MATLAB/Simulink* [117, 114], implementiert. Weiterhin wurde der durchgehende Einsatz des SES/MB Frameworks zum Variantenmanagement, zunächst nur eingeschränkt auf das Modellmanagement diskutiert, wobei das Variantenmanagement in folgende Phasen: (i) *Variantenanalyse*, (ii) -*formalisierung*, (iii) -*implementierung* und (iv) -*generierung* unterteilt wurde.

Im vierten Kapitel wurden die erzielten Ergebnisse auf die Anforderungen simulationsbasierter Experimente erweitert. Aufbauend auf dem modular-hierarchischen Ansatz des SES/MB Frameworks und unter Nutzung des Konzepts des Experimental Frame wurde eine neue Strukturierung für simulationsbasierte Experimente entwickelt. Dabei wurde gemäß der eingeführten Klassifikation der Fokus auf die Klasse der komplexen Experimente gelegt. Ein komplexes Simulationsexperiment besteht aus drei miteinander gekoppelten Komponenten: (i) einer *Experimentmethode*, (ii) der *Simulationsmethode* und (iii) einem *Simulationsmodell*. Die Experimentmethode ist ein softwaretechnisch umgesetztes numerisches Verfahren, beispielsweise vom Typ Screening, Sensitivitätsanalyse oder Optimierung. Die Simulationsmethode bildet die Schnittstelle zwischen dem Simulationsmodell, der Experimentmethode und dem Simulator, während die unmittelbare Ausführung eines Simulationslaufes durch den Simulator erfolgt. Auf Basis dieser Strukturierung wurde das Variantenmanagement komplexer Simulationsexperimente an einem Beispiel vorgestellt. Es konnte gezeigt werden, dass das zuvor konkretisierte SES/MB Framework ohne zusätzliche Modifikationen zum Variantenmanagement komplexer simulationsbasierter Experimente anwendbar ist, wenn in der MB neben konfigurierbaren Modellkomponenten auch konfigurierbare Experimentkomponenten organisiert sind. Analog zum klassischen SES/MB

Framework für die Modellebene muss der Anwender bis zu diesem Punkt manuell entscheiden, welche Varianten von Simulationsexperimenten in welcher Reihenfolge zu generieren und auszuführen sind.

Zur angestrebten Automatisierung der Generierung und Ausführung von unterschiedlichen Simulationsexperimentvarianten erfolgte im fünften Kapitel die Entwicklung eines Konzeptrahmens für das automatisierte Experimentieren. Der Konzeptrahmen gliedert sich in: (i) *Experimentsteuerung*, (ii) *SES/MB Framework* und (ii) *Experimentausführungseinheit*. Die Experimentsteuerung beschreibt, welche Varianten von Simulationsexperimenten in welcher Reihenfolge und unter welchen Bedingungen zu generieren sind. Das konkretisierte SES/MB Framework, welches eine Inputschnittelle definiert, codiert mit der SES und der MB alle möglichen Simulationsexperimentvarianten und führt auf Anweisung der Experimentsteuerung eine konkrete Variantengenerierung durch. Die Experimentausführungseinheit übernimmt die Experimentausführung. Da umfangreiche Experimente, sogenannte Simulationsstudien, die oft verschiedene komplexe simulationsbasierte Experimente umfassen, schnell zu langen Rechenzeiten führen können, wurden verschiedene Ansätze zur parallelen/verteilten Ausführung von Simulationsexperimenten betrachtet.

Der Konzeptrahmen ist generisch für komplexe simulationsbasierte Experimente und die Unterklasse der einfachen simulationsbasierten Experimente anwendbar. Für die Klasse der hochkomplexen simulationsbasierten Experimente zeigte sich, dass eine Adaption des Konzeptrahmens sinnvoll ist. Diese wurde konkret am Beispiel einer simulationsbasierten Struktur- und Parameteroptimierung aufgezeigt.

Im sechsten Kapitel wurde die entwickelte Methodik zum Variantenmanagement an einem Beispiel aus dem Anwendungsbereich der Produktion und Logistik vorgestellt. Das übergeordnete Ziel war die Identifikation einer optimalen fertigungstechnischen Prozesskettenvariante. Dazu wurden insgesamt 486 Varianten der Klasse komplexe Simulationsexperimente in einer SES spezifiziert sowie die notwendigen Modell- und Experimentkomponenten implementiert und in einer MB organisiert. Die 486 Varianten folgen aus 81 unterschiedlichen Varianten fertigungstechnischer Prozessketten mit zugehörigem Parameterraum und deren Untersuchung mit jeweils zwei unterschiedlichen Experimentmethoden vom Typ Screening, Sensitivitätsanalyse sowie Optimierung. Bei diesen komplexen simulationsbasierten Experimenten müssen eine erhebliche Anzahl von Simulationsläufen pro einzelnes Experiment durchgeführt werden. Es zeigte sich, dass bei dieser Problemstellung auf der verfügbaren PC-Standardhardware die Ausführungszeit eines Simulationslaufes bis zu fünf Minuten betragen kann, woraus bei einer rein sequentiellen Ausführung eine enorm lange Gesamtausführungszeit für die Untersuchung aller Varianten folgt. Der Grund für die langen Ausführungszeiten liegt im hohen Detaillierungsgrad einzelner Prozesskettenkomponenten auf Modellebene. Der Detaillierungsgrad folgte aus der Zielstellung der Untersuchung, einer Produktionsplanung unter Berücksichtigung einer detaillierten Bestimmung der zeitabhängigen Energieverbräuche der einzelnen Fertigungsoperationen. Da zum Experimentieren nur eine Standardhardware zur Verfügung stand, konnte im Rahmen dieser Arbeit nur das Screening unter Verwendung von einer Experimentmethode vollständig ausgeführt werden. Die damit gewonnenen Simulationsergebnisse wurden anschließend einer gewöhnlichen Datenanalyse unterzogen. Dabei zeigte sich, dass bereits mit dieser Untersuchung suboptimale Lösungen abgeleitet werden konnten. Auch wenn keine vollständige Untersuchung des Anwendungsbeispiels aufgrund nicht verfügbarer parallelen/verteilter Hardware möglich

war, konnte der theoretische Ansatz der entwickelten Methodik, mit Ausnahme der parallelen/verteilten Ausführung von Experimenten, praxisnah anhand der durchgeführten Teiluntersuchung des Anwendungsbeispiels aufgezeigt werden.

Mit der stetigen Entwicklung der Rechentechnik, insbesondere bei den Multicore-Prozessoren, werden derartige Problemstellungen vermutlich auf zukünftiger Standardrechentechnik lösbar sein. Für den praktischen Einsatz der entwickelten Methode sind weitere Anwendungsbeispiele zu untersuchen.

Die Spezifikation der SES, die Implementierung von konfigurierbaren Komponenten sowie deren Organisation in einer MB und die Spezifikation sowie Implementierung der Experimentsteuerung bedingt in der Regel einen nicht unerheblichen zeitlichen und qualifizierten personellen Aufwand. Es ist im Einzelfall zu klären, ab welchem Variantenumfang sich der Mehraufwand eines automatisierten Variantenmanagements gegenüber der klassischen Herangehensweise in der M&S rentiert.

Im Zuge dieser Arbeit wurde die umfangreiche Thematik der *Verifikation, Validierung und Testing (VV&T)* ausgelassen. Für die Vervollständigung der erarbeiteten Methoden sollten auch Ansätze aus dem Bereich der VV&T in das Variantenmanagement integriert werden.

Wie in der Arbeit erwähnt, besitzt die Thematik des Variantenmanagements auch in anderen Domänen, wie zum Beispiel dem *Software Product Line Engineering*, eine hohe Relevanz. Aus der Sicht des Autors wäre es sehr interessant, die entwickelte Methodik aus der Perspektive des allgemeinen Software Engineerings zu analysieren und eine Übertragung auf dort zu lösende Problemstellungen zu prüfen.

Literaturverzeichnis

[1] Alba, E., Troya, J.M.: A survey of parallel distributed genetic algorithms", COMPLEXITY, 4(4), Wiley Periodicals, Inc., p. 3-72, 1999

[2] Alt, O.: Modellbasierte Systementwicklung mit SysML, Carl Hanser Verlag München, 2012

[3] Apel, A., Kästner, C.: An Overview of Feature-Oriented Software Development, Journal of Object Technology, 8(5), ETH Zürich, pp. 49-84, 2009

[4] Baez-Lopez, D.: MATLAB: With Allpications to Engineering, Physics and Finance, CRC Press, Taylor & Francis Group, 2010

[5] Balci, O.: Verification, Validation and Testing, In: Banks, J. (ed.): Handbook of Simulation, John Wiley & Sons Inc., pp. 335-393, 1998

[6] Balci, O.: Verification, Validation and Accreditation of Simulation Models, In: Winter Simulation Conference 1997, 7-10 December, Atlanta, GA, USA, pp. 135-141, 1997

[7] Balzert, H.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering, 3. Auflage, Spektrum Akademischer Verlag, 2009

[8] Banks, J., Carson, J., Nelson, B., L., Nicol, D.: Discrete-Event Systems Simulation, 4th Edition, Pearson, 2005

[9] Barton, R. R.: Designing Simulation Experiments, In: 2013 Winter Simulation Conference, 8-11 Dec., Washington, D. C., USA, pp. 342-353, 2013

[10] Batory, D.: Feature Models, Grammars, and Propositional Formulas, In: Software Product Lines, 9th International Conference, SPLC 2005, 26.-29. September, Rennes, France, pp. 7-20, 2005

[11] Bhatnagar, S., Prasad, H., L., Prashanth, L., A.: Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods, Springer-Verlag, 2013

[12] Bock, B., Schmitz, M., Wenzel, S.: Entwicklung von Methodiken zur systematischen Planung von Experimenten in Simulationsstudien, In: Zülch, G., Stock, P., (Hrsg.), Integrationsaspekte der Simulation: Technik, Organisation und Personal, KIT Scientific Publishing 2010, pp. 493-500, 2010

[13] Bolado-Lavin, R., Badea, A. C.: Review of Sensitivity Analysis Methods and Experience for Geological Disposal of Radioactive waste and Spent Nuclear Fuel, JRC Scientific and Technical Reports, Luxembourg: Office for Official Publications of the European Communities, 2008

[14] Burotzky, W.: Bond Graph Methodology: Development and Analysis of Multidisciplinary Dynamic Systems Models, Springer Verlag, 2010

[15] Campolongo, F., Kleijnen, J. P. C., Andres, T.: Screening Methods, In: Saltelli, A., Chan, K., Scott, E. M. (Hrsg.), Sensitiviy Analysis, John Wiley & Sons, Ltd., pp.65-80, 2000

[16] Carson, Y., Maria, A.: Simulation Optimization: Methods and Applications, In: 1997 Winter Simulation Conference, 7. – 10. December, Atlanta, GA, USA, pp. 118-126, 1997

[17] Cavarlé, G., Plantec, A., Costiou, S., Ribuad, V.: Dynamic Round-Trip Engineering in the context of FOMDD, In: International Workshop on Smalltalk Technologies, Prague, Czech Republic, 22-26 August, 7 pages, 2016

[18] Cellier, F., Kofman, E.: Continuous System Simulation, Springer Verlag, 2006

[19] Cetinkaya, D., Verbraeck, A., Seck, M.D.: MDD4MS: A Model Driven Development Framework for Modeling and Simulation, In: 2011 Summer Computer Simulation Conference, 27-30 June, Hague, Netherlands, pp. 113-121, 2011

[20] Chan, K., Tarantola, S., Saltelli, A., Sobol, I. M.: Variance-Based Methods, In: Saltelli, A., Chan, K., Scott, E. M. (eds.), Sensitiviy Analysis, John Wiley & Sons, Ltd., pp.167-197, 2000

[21] Clymer, J. R.: Simulation-Based Engineering of Complex Systems, John Wiley & Sons, Inc., 2009

[22] Cooke, R., M., van Noortwijk, J., M.: Graphical Methods, In: Saltelli, A., Chan, K., Scott, E. M. (eds.), Sensitiviy Analysis, John Wiley & Sons, Ltd., pp. 245-264, 2000

[23] Cordes, P., Schmitz, P.: Prolog: Eine met hodische Einführung, Vieweg+Teubner Verlag, 1990

[24] Damyanov, I., Sukalinska, M.: Domain Specific Languages in Practice, International Journal of Computer Applications (0975 – 8887), 115(2), 2015

[25] Davis, P.K., Bigelow, J.H.: Experiments in Multiresolution Modeling, Published by Research and Development (RAND), National Defense Reasearch Institute, Santa Monica, CA, 1999

[26] Dean, A., Lewis S.: Screening: Methods for Experimentation in Industry, Drug Discovery and Genetics, Springer Science Business Media Inc., 2006

[27] Dearle, F.: Groovy for Domain-specific Languages – Second Edition, packt Publishing Ltd., 2015

[28] Deckert, A.: Simulationsbasierte Optimierung einer agentenbasierten Simulation mit Anwendungen zur Preisoptimierung im Mobilfunk, Diss. Universität Augsburg, 2012

[29] Denker, M.: Einführung in die Analysis dynamischer Systeme, Springer-Verlag Berlin Heidelberg, 2005

[30] Dupuy, D., Corre, B., Claeys-Bruno, M., Sergent, M.: Comparison of different screening methods, Case Studies in Business, Industry & Government Statistics 5(2), pp. 115-125, 2014

[31] Durak, U., Pawletta, T., Oguztuzun, H.: System Entity Structure and Model Base Framework in Model Based Engineering of Simulation For Technical Systems, In: 2017 Spring Simulation Multi-Conference, 23-26 April, Virginia Beach, VA, USA, 10 pages, 2017

[32] Durak, U., Pruter, I., Gerlach, T. et al.: Using System Entity Structure to Model the Elements of Scenario in a Research Flight Simulator, AIAA Modeling and Simulation Technologies Conference, 9-13 January, Grapevine, Texas, USA, 10 pages, 2017

[33] Dustdar, S., Gall, H., Hauswirth, M.: Software-Architekturen für Verteilte Systeme, Springer-Verlag, 2003

[34] Dziobek, C., Przytas, W., Loew, J., Weiland, J.: Handling Functional Variants in Simulink Models, Online-Quelle: https://www.pure-systems.com/fileadmin/downloads/articles/P.33-37_EK-Automotive_2-08-en.pdf, 5 pages, Zugriff: 29.04.2017

[35] Ekström, P.-A., Broed, R.: Sensitivity Analysis Methods and a Biosphere Test Case Implemented in EIKOS, Working Report 2006-31, FI-27160 OLKILUOTO Finland, 2006

[36] Ekström, P.-A.: EIKOS: A Simulation Toolbox for Sensitivity Analysis, Degree Project, February 2005, Faculty of Science and Technology, Uppsala University, 2005

[37] Eley, M.: Simulation in der Logistik: Eine Einführung in die Erstellung ereignisdiskreter Modelle unter Verwendung des Werkzeuges „Plant Simulation“, Springer Verlag, 2012

[38] Figueira, G., Almada-Lobo, B.: Hybrid Simulation Optimization Methods: A Taxonomy and Discussion, Simulation Modelling Practice and Theory 46, pp. 118-134, 2014

[39] Fink, A., Rothlauf, F.: Heuristische Optimierungsverfahren in der Wirtschaftsinformatik, Working Paper 10/Lehrstuhl für ABWL und Wirtschaftsinformatik, Online-Quelle:http://wi.bwl.uni-mainz.de/Dateien/wi-metaheuristics-working-paper10_06.pdf, pp. 1-15, 2006, Zugriff: 10.08.2016

[40] Fink, R., Pawletta, S., Pawletta, T.: SCE based Parallel Processing and Applications in Simulation, SNE Special Issue: Parallel and Distributed Simulation Methods and Environments, 6(2), pp. 37-50, 2006

[41] Fink, R.: Untersuchungen zur Parallelverarbeitung mit Wissenschaftlich-technischen Berechnungsumgebungen, Diss. Universität Rostock, 2007

[42] Foures, D., Albert, V., Nketsa, A.: A new specification-based qualitative metric for simulation model validity, Simulation Modelling Practice and Theory 66, Elsevier, pp. 1-15, 2016

[43] Foures, D., Albert, V., Nketsa, A.: Simulation validation using the compatibility between Simulation Model and Experimental Frame, In: 45th Summer Simulation Multi-Conference (SummerSim’13), 7.-10. July, Toronto, ON, Canada, pp.75-82, 2013

[44] Frey, H.,C., Patil, S., R.: Identification and Review of Sensitivity Analysis Methods, Risk Analysis 22(3), pp. 553-578, 2002

[45] Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language, 3rd Edition, Elsevier Verlag, 2015

[46] Fu, M.C.: Optimization for Simulation: Theory vs. Practive, INFORMS Journal on Cuputing 14(3), INFORMSPubs, pp. 192-215, 2002

[47] Fujimoto, R.M.: Parallel and Distributed Simulation Systems, John Whiley & Sons, Inc, 2000

[48] Fujimoto, R.M.: Parallel and Distributed Simulation, In: Winter Simulation Conference 2015, 6. – 9. December, Huntington Beach, CA, USA, pp. 45-59, 2015

[49] Gehlsen, B.: Automatisierte Experimentplanung im Rahmen von Simulationsstudien, Diss., Uni. Hamburg, 2004

[50] Glöckler, M.: Simulation mechatronischer Systeme: Grundlagen und technische Anwendung, Springer-Vieweg, 2014

[51] Goldstein, R., Breslav, S., Khan, A.: A quantum of continuous simulated time, In: Symposium on Theory of Modeling & Simulation (TMS-DEVS'16), 3.-6. April, Pasadena, CA, USA, 8 pages, 2016

[52] Gotika, P., Mahapatra, S.: Design Variant Management in Model-Based Design, In: AIAA Modeling and Simulation Technologies Conference, AIAA SciTech Forum, 5.-9. January, 2015, Kissimmee, FL, USA, 13 pages, 2015

[53] Gronniger, H., Krahn, H., Pinkernell, C., Rumpe, B.: Modeling Variants of Automotive Systems using Views, In: Modellierungs-Workshop MBEFF: Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen, 12-14. März 2008, Berlin, Deutschland, 14 Seiten, 2008

[54] Gruber, T.R.: A Translation Approach to portable Ontology Specifications, Knowledge Acquisition, 5(2), 27 pages, 1993

[55] Günther, M., Velten, K.: Mathematische Modellbildung und Simulation: Eine Einführung für Wissenschaftler, Ingenieure und Ökonomen, WILEY-VCH Verlag GmbH & CO. KGaA, 2014

[56] Gutekunst, B., Weiland, J.: Handbuch von Varianz in Simulink aus funktionsorientierter Sicht, In: INFORMATIK 2011 41. Jahrestagung der gesellschaft für Informatik, 4. – 7. Oktober, 2011, Berling, GER, 13 pages, 2011

[57] Haber, A., Kolassa, P., Manhart, P., Nazari, P., Rumpe, B., Schaefer, I.: First-Class Variability Modeling in Matlab/Simulink, In: the 7th International Workshop on Variability Modeling of Software-intensive Systems, 23.-25 January 2013, ACM, NY, USA, pp. 11-18, 2013

[58] Hagendorf, O.: Simulation Based Parameter and Structure Optimiesation of Discrete Event Systems, Diss. Liverpool John Moores University, 2009

[59] Halmans, G., Pohl, K,: Modellierung der Variabilität einer Software-Produktlinie, In: Modellierung in der Praxis – Modellierung für die Praxis, Arbeitstagung der GI, 25.-27. März 2002 in Tutzing, pp. 63- 74, 2002

[60] Han, J., Miller, J.A., Silver, G.A.: STOP: Ontology for Simulation Optimization for Scientific Experiments, In: the 2011 Winter Simulation Conference, 11.-14. December, Phoenix, AZ, USA, pp. 2914-2925, 2011

[61] Han, S.-O.: Varianzbasierte Sensitivitätsanalyse als Beitrag zur Bewertung der Zuverlässigkeit adaptorischer Struktursysteme, Diss. TU Darmstadt, 2011

[62] Hare, W., Nutini, J., Tesfamariam, S.: A survey of non-gradient optimization methods in structural engineering, Advance in Engineering Software 59, pp. 19-28, 2013

[63] Harel, D.: Statecharts: A visual formalism for complex systems, In: Science of Computer Programming 8, pp. 231-274, 1987

[64] Heitmann, K.: Sichere Prognosen für die Produktionsoptimierung mittels stochastischer Modelle, Diss. TU München, 1999

[65] Helmerich, A., Koch, N., Mandel, L., et al.: Study of Worldwide Trends and R&D Programmes in Embedded Systems in View of Maximising the Impactofa Technology Platformin the Area, Final Report for the European Commision, Brussels Belgium, 2005

[66] Himmelspach, J., Uhrmacher A.M.: Plug'n Simulate, In: the 40th Annuals Simulation Symposium (ANSS'07), 26-28 März, Washington, DC, USA, pp. 137-143, 2007

[67] Holdschick, H., Commerell, W.: Variantenmanagement in der modellbasierten Produktentwicklung von Fahrzeugsystemen, In: ASIM STS/GMMS 2013, Düsseldorf, ARGESIM Report 41, ASIM Mitteilungsnummer AM 145, ARGESIM/ASIM Pub. TU Vienna, Austria, 28.2. – 1.3., pp. 111-118, 2013

[68] Holdschick, H.: Beherrschung dynamischer Variantenmodelle für eingebettete Fahzeugsysteme, Diss. Eberhard Karls Univ. Tübingen, 2014

[69] Hussmann, H., Meixner, G., Zuehlke, D.: Model-Driven Development of Advanced User Interfaces, Springer Verlag, 2010

[70] Iooss, B., Lemaître, P.: A review on global sensitivity analysis methods, In: Meloni, C., Deliino, G., Uncertainty management in Simulation-Optimization of Complex Systems: Algorithms and Applications", Springer, pp. 101-122, 2015

[71] Jaensch, M.: Modulorientiertes Produktlinien Engineering für den modellbasierten Elektrik/Elektronik-Architekturentwurf, Diss. Karlsruher Institut für Technologie (KIT), 2012

[72] Jézéquel, J.-M.: Model-Driven Engineering for Software Product Lines, ISRN Software Engineering, vol. 2012, article ID 670803, 24 pages

[73] Kamga, J., Herrmann, J., Joshi, P.: Deployment of model-based technologies to industrial testing, Deliverable: D-MINT automotive case study – Daimler, Deliverable 1.1, ITEA2 Project, 2007

[74] Kang, K., C., Cohen, S., G., Hess, J., A., Novak, W., E., Peterson, A., S.: Feature-Oriented Domain Analysis Feasibility Study, Technical Report, CMU/SEI-90-TR-21, ESD-90-TR-222, Software Engineering Institute, Carnegie Mello University, 1990

[75] Karatas, A. S., Oguztüzün, H., Dogru, A.: From extended feature models to constraint logic programming, Science of Computer Programming 78, Elsevier, pp. 2295-2315, 2013

[76] Kelton, W.,D., Barton, R., R.: Experimental Design for Simulation, In: the 2003 Winter Simulation Conference, 7.-10. December, New Orleans, LA, USA, pp. 59-65, 2003

[77] Kim, B.S., Lee, S.J., Kim, T.G., Song, H.S.: MapReduce based Experimental Frame for Prallel and Distributet Simulation Using Hadoop Platform, In: 28th European Conference on Modelling and Simulation, 27.-30. May, Brescia, Italy, 6 pages, 2014

[78] Kim, T.: Ontology/Data Engineering Based Distributed Simulation Over Service Oriented Architecture for Network Behavior Analysis, Diss. of Uni. of Arizona, 2008

[79] Kleijnen, J. P. C.: Design and Analysis of Simulation Experiments, Springer Science, 2008

[80] Kleijnen, J.P.C.: Experimental Design for Sensitivity Analysis, Optimization and Validation of Simulation Models, In: Banks, J. (ed.), Handbook of Simulation, John Wiley & Sons Inc., pp. 173-223, 1998

[81] Kleppmann, W.: Versuchsplanung: Produkte und Prozesse optimieren, 8. Auflage, Carl Hanser Verlag, 2013

[82] Köchel, P., Riedel, M.: Performance Evaluation of Parallel Genetic Algorithms for Optimization Problems of Different Conplexity, In: Joubert, G.R., Nagel, W.E., Peters, F.J., Walter, W.V. (eds), Parallel Computing: Software Technology, Algorithms, Architectures and Applications, Elsevier B.V., p. 313-320, 2004

[83] Kolassa, C., Rendel, H., Rumpe, B.: Evaluation of Variability Concepts for Simulink in the Automotive Domain, In: Proc. of 48th Hawaii International Conference on System Sciences (HICSS), 5.-8. January 2015, Kauai, HI, USA, pp. 5373-5382, 2015

[84] Kolitz, K.: Systemdesign im Market-Engineering – Experimente zu Teilnehmerverhalten und Technologieakzeptanz, Diss. Universität Fridericiana zu Karlsruhe (TH), 2007

[85] Kruse, R., Borgel, C., Braune, C., Klawonn, F., Moewes, C., Steinbrecher, M.: Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes Netze, 2. Auflage, Springer Vieweg, 2015

[86] Kunz, G. Parallel Discrete Event Simulation, In: Wehrle, K., Günes, M., Gross, J., (Hrsg.): Modeling and Tools for Network Simulation, Springer-Verlag, 2010

[87] Larek, R.: Ressourceneffiziente Auslegung von fertigungstechnischen Prozesskette durch Simulation und numerische Optimierung, Diss. Universität Bremen, 2012

[88] Lattner, A.D., Bongo, T., Lorion, Y., Timm, I.J.: A Knowledge-Based Approach to Automated Simulation Model Adaption, In: Spring Simulation Conference 2010, 11-15 April, Orlando, Florida, pp.200-207, 2010

[89] Lattner, A.D., Pitsch, H., Timm, I.J., Spieckermann, S., Wenzel, S.: AssistSim – Towards Automation of Simulation Studies in Logistics, Simulation Notes Europe 21(3- 4), ARGESIM/ASIM Pub., Vienna Univ. of Technology, Austria, pp. 119– 128, 2011

[90] Lattner, A.D.: Towards Automation of Simulation Studies – Artificial Intelligence Methodologies for the Control and Analysis of Simulation Experiments, Habilitationsschrift Universität Trier, 2012

[91] Law, A.M., Kelton, W.D.: Simulation, Modeling and Analysis, The McGraw-Hill Companies, Inc., 3th Edition, 2000

[92] Leye, S.: Towards Guiding Simulation Experiments, Diss. Universität Rostock, 2013

[93] Liebl, F.: Simulation: Problemorientierte Einführung, R. Oldenbourg Verlag München Wien, 1995

[94] Lingnau, V.: Variantenmanagement: Produktionsplanung im Rahmen einer Produktdifferenzierungsstrategie, Erich Schmidt Verlag, 1994

[95] Lobo, F., G., Lima, C., F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms, Springer-Verlag, 2007

[96] Loper, M. L.: The Modeling and Simulation Life Cycle Process, In: Loper, M., L., (eds.) Modeling and Simulation in the Systems Engineering Life Cycle, Springer-Verlag London, pp. 17-27, 2015

[97] Lunze, J.: Ereignisdiskrete Systeme: Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen, Oldenburg Verlag, 2006

[98] Luque, G., Alba, E.: Parallel Genetic Algorithms: Theory and Real World Applications, Springer-Verlag, 2011

[99] März, L., Krug, W., Rose, O., Weigert, G.: Simulation und Optimierung in Produktion und Logistik: Praxisorientierter Leitfaden mit Fallbaspielen, Springer-Verlag, 2011

[100] März, L., Krug, W.: Kopplung von Simulation und Optimierung", In [99], pp.41-63, 2011

[101] Meinel, C., Mundhenk, M.: Mathematische Grundlagen der Infromatik: Mathematisches Denken und Beweisen: Eine Einführung, 6. Auflage, Springer-Verlag, 2015

[102] Mittal, S., Martin, J.L.R.: Netcentric System of Systems Engineering with DEVS Unified Processes, CRC Press, 2013

[103] N.N.: Industrie 4.0: Innovationen für die Produktion von morgen, Bundesministerium für Bildung und Forschung: Referat Forschung für Produktion, Dienstleistung und Arbeit, BMBF, 2015

[104] Nader, B., Filippi, J.,B., Basgambiglia, P.A.: An Experimental Frame for the Simulation of Forest Fire Spread, In: 2011 Winter Simulation Conference, 11.-14. December, Phoenix, AZ, USA, pp. 1010-1022, 2011

[105] Nauheimer, H., Bertsche, B., Lechner, G.: Fahrzeuggetriebe: Grundlagen, Auswahl, Auslegung und Konstruktion. 2. Auflage. Springer, 2007

[106] Ndih, E.D.N., Cherkaoui, S.: Simulation methods, techniques and tools of computer systems and networks, In: Obaidat, M.S., zarai, F., Nicopolitidis, P., (eds.): Modeling and Simulation of Computer networks and Systems, Publisher: Morgan Kaufmann, p. 485-504, 2015

[107] Neyrick, A., Lechner, A., Verl, A.: Automatic Variant Configuration and generation of Simulation Models for Comparison of Plant and Machinery Variants, Procedia CIRP, vol. 29, pp. 62-67, 2015

[108] Nguyen, A.-T., Reiter, S., Rigo, P.: A review on simulation-based optimization methods applied to building performance analysis, Applied Energy vol. 113, pp. 1043-1058, 2014

[109] Nunes, D., Antunes, L,: Parallel Execution of Social Simulation Models In a grid Environment, In: Giardini, F., Amblard, F., (Hrsg.): Multi Agend-Based Simulation XIII, MABS 2012. Lecture botes in Computer Science, vol. 7838, Springer, pp. 115-129, 2013

[110] Oster, S.: Feature Model-Based Software Product Line Testing, Diss. TU Darmstadt, 2011

[111] Papageorgiou, M., Leibold, M., Buss, M.: Optimierung: Statistische, dynamische, stochastische Verfahren für die Anwendung, Springer-Verlag, 4. Auflage, 2015

[112] Pawletta T., Schmidt A., Junglas P.: A Multimodeling Approach for the Simulation of Energy Consumption in Manufacturing. SNE - Simulation Notes Europe 27(2), ARGESIM Pub., Vienna, Austria, 2017, pp. 115-124.

[113] Pawletta, S.: Erweiterung eines wissenschaftlich-technischen berechnungs- und Visualisierungssystems zu einer Entwicklungsumgebung für parallele Applikationen, Fortschrittsberichte Simulation Band 7, ARGESIM/ASIM-Verlag, Wien, 2000

[114] Pawletta, T., Pascheka, D., Schmidt, A., Pawletta, S.: Ontology-Assisted System Modeling and Simulation within MATLAB/Simulink. SNE Simulation Notes Europe, ARGESIM/ASIM Pub. TU Vienna, Austria, 24(2)-8/2014, 59-68 (DOI: 10.11128/sne.24.tn.102241), 2014

[115] Pawletta, T., Schmidt, A., Zeigler, B.P., Durak, U.: A Framework for Metamodeling of Multi-Variant Systems and Reactive Simulation Model generation, In:23th Symposium Simulationstechnik (ASIM2016), 7 – 9 September, Dresden, GER, pp. 131-138, 2016

[116] Pawletta, T., Schmidt, A., Zeigler, B.P., Durak, U.: Extended Variability Modeling Using System Entity Structure Ontology within MATLAB/Simulink, In: 49th Annual Simulation Symposium, 3 – 6 April, Pasadena, CA, USA, pp. 22:1-22:8, 2016

[117] Pawletta, T., Schwatinski, T., Schmidt, A., Pascheka, D.: SES Toolbox for MATLAB/Simulink, Research Group CEA, Univ. Wismar, Online-Quelle: https://fiw.hs-wismar.de/fakultaet/ausstattung-forschung/

forschungsgruppen/research-group-cea/software-projects/ses-toolbox/ , Zugriff: 29.07.2018

[118] Pawlikowski, K., Yau, V.W.C., McNickle, D.: Distributed Stochastic Discrete-Event Simulation in Parallel Time Stream, In: Winter Simulation Conference 1994, 11. – 14. December, Conorado, CA, USA, p. 723-730, 1994

[119] Peng, D.: Reusing Simulation Experiments for Model Configuration and Extension, Diss. Universität Rostock, 2017

[120] Philipp, K.: Experimentelles Denken: Theoretische und empirische Konkretisierung einer mathematischen Kompetenz, Diss. Pädagogische Hochschule Freiburg, 2012

[121] Pohl, K., Blöckle, G., van der Linden, F.: Software Product Line Engineering: Foundation, Principles and Techniques, Springer-Verlag, 2005

[122] pure-systems GmbH: pure::variants for Simulink, Online-Quelle: https://www.pure-systems.com/products/pure-variants-for-simulink-288.html, Zugriff: 21.07.2018

[123] pure-systems GmbH: pure::variants User's Guide, Version 4.0.0.685 for pure::variants 4.9, 2016, Online-Quelle: http://www.pure-systems.com/fileadmin/downloads/pure-variants/doc/pv-user-manual.pdf, Zugriff: 23.07.2018

[124] Rabe, M., Spieckermann, S., Wenzel, S.: Verifikation und Validierung für Simulation in Produktion und Logistik: Vorgehensmodell und Techniken, Springer Verlag, 2008

[125] Reisig, W.: Petrinetze: Modellierungstechnik, Analysemethode, Fallstudien, Vieweg Verlag, 2010

[126] Röhl, M., König-Ries, B., Uhrmacher, A.M.: An Experimental Frame of Evaluating Service Trading in Mobile ad-hoc Networks, In: 2nd conference of GI-Fachgruppe: Mobilität und Mobile Informationssysteme, 6th March, Aachen, Germany, pp. 37-48, 2007

[127] Roßner, T., Brandes, C., Götz, H., Winter, M.: Basiswissen Modellbasierter Test, dpunkt.verlag, 2010

[128] Rozenblit, J.W., Zeigler, B.P.: Concepts for knowledge-based system design environment, In: Winter Simulation Conference 1985, 11-13 December, San Francisco, CA, USA, pp. 223 – 231, 1985

[129] Rybaki, S.: Towards Reproducible Simulation Studies with JAMES II, Diss. Uni. Rostock, 2016

[130] Ryssel, U.: Automatische Generierung von feature-orientierten Produktlinien aus Varianten von funktionsblockorientierten Modellen, Diss. TU Dresden, 2014

[131] Saltelli, A., Ratto, M., Andres, T., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis: The Primer, John Wiley & Sins, Ltd., 2008

[132] Saltelli, A.: What is Sensitivity Analysis? In: Saltelli, A., Chan, K., Scott, E. M. (eds.), Sensitiviy Analysis, John Wiley & Sons, Ltd., pp. 3-13, 2000

[133] Santucci, J.-F., Capocchi, L., Zeigler, B.P.: System entity structure extension to time granularity into DEVS modeling and simulation, SIMULATION: Transaction of the

Society for Modeling and Simulation International, 92(8), SAGE Publication, pp.747-769, 2016

[134] Sargent, R., G.: Verification and Validation of Simulation Models, In: Winter Simulation Conference 2011, 11-14 December, Phoenix, AZ, USA, p. 183-198, 2011

[135] Sauerbier, T.: Theorie und Praxis von Simulationssystemen, Springer Verlag, 1999

[136] Schäuffele, J., Zurawka, T.: Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen, 6. Auflage, Springer Fachmedien Wiesbaden, 2016

[137] Schlott, S.: Wahnsinn mit Methode, Automobil Produktion 01, pp. 38-42, 2005

[138] Schmidt, A.: Entwicklung eines Simulators zur energetischen Bewertung von Prozessketten der spanenden Bauteilfertigung mit MATLAB/SimEvents, Master-Thesis, FIW/MVU der Fachhochschule Wismar, 2012

[139] Schmidt, A., Durak, U., Pawletta, T.: Model-Based Testing Methodology Using System Entity Structure for MATLAB/Simulink models, Simulation: Transaction of the Society for Modeling and Simulation International, 92(8), Sage Pub., pp.729-746, 2016

[140] Schmidt, A., Pawletta, T. and Junglas P.: A Layered Structure for Modeling Manufacturing Processes with the Inclusion of Energy Consumption. In: ASIM-Treffen STS/GMMS, Lippstadt, Germany, 10./11., March, 2016, ARGESIM Report 51 & ASIM Mitteilung AM 158, ARGESIM Pub. Vienna/Austria, 2016, pp. 155-164.

[141] Schmidt, A., Pawletta, T.: Ein Ontologie-basierter Modellierungs- und Simulationsansatz am Beispiel der ressourceneffizienten Planung spanender Prozessketten, In: 15. ASIM Fachtagung Simulation in Produktion und Logistik, Paderborn, 9.-11. Oktober 2013, HNI-Verlagsschriftenreihe Bd. 316, pp. 481-490

[142] Schmidt, A., Pawletta, T.: Hybride Modellierung fertigungstechnischer Prozessketten mit Energieaspekten in einer ereignisorientierten Simulationsumgebung, In: ASIM 2014 - 22. Symposium Simulationstechnik, Berlin, 03./05.09.2014, ARGESIM Report 43, ASIM Mitteilung AM 151, ARGESIM/ASIM Pub. Vienna, Austria, 2014, p. 109-116.

[143] Schmidt, A., Pawletta, T.: Ontologische Modellierung und ereignisorientierte Simulation mit MATLAB/SimEvents am Beispiel der ressourceneffizienten Prozesskettenplanung, In: ASIM STS/GMMS 2013, Düsseldorf, ARGESIM Report 41, ASIM Mitteilungsnummer AM 145, ARGESIM/ASIM Pub. TU Vienna, Austria, 28.2. – 1.3. 2013, pp. 65-74

[144] Schrems, S.: Methoden zur modellbasierten Integration des maschinenbezogenen Energiebedarfs in die Produktionsplanung, Diss. TU Darmstadt, 2014

[145] Schulz, R.: Parallele und verteilte Simulation bei der Steuerung komplexer Produktionssysteme, Diss. TU Illmenau, 2002

[146] Schuster, N.: Coordinating Service Compositions: Model and Infrastructure for Collaborative Creation of Electronic Documents, Diss. Karlsruher Institut für Technologie (KIT), 2013

[147] Schützel, J., Peng, D., Uhrmacher, A., M., Perrone, L.F.: Perspectives on languages for specifying simulation experiments, In: the 2014 Winter Simulation Conference, 7.-10 December, Savannah, GA, USA, pp. 2836-2847, 2014

[148] Schwatinski, A., Pawletta, T.: Ontologische Modellierung und Modellgenerierung in der MATLAB/Simulink Umgebung: Die Tiny SES Toolbox, In: ASIM STS/GMMS 2013, Düsseldorf, ARGESIM Report 41, ASIM Mitteilungsnummer AM 145, ARGESIM/ASIM Pub. TU Vienna, Austria, 28.2. – 1.3., pp. 57-64, 2013

[149] Schwatinski, T., Pawletta, T., Pawletta, S.: Flexible Task-oriented Robot Control Using the System Entity Structure and Model Base Approach, Simulation Note Europe, 22(2), pp. 107-114, 2012

[150] Scilab Enterprises: Scilab for very beginners, Scilab Enterprises, Online-Quelle, https://www.scilab.org/content/download/849/7901/file/Scilab_beginners.pdf, Zugriff: 30.07.2018

[151] Soykan, B.: A hybrid Tabu/Scatter Search Algorithm for Simulation-Based Optimization of Multi-Objective Runaway Operations Scheduling, Diss. Old Dominion University, 2016

[152] Sree-Kumar, A., Planas, E., Clarisó, R.: Analysis of Feature Models Using Alloy: A Survey, In: 7th International Workshop on Formal Methods and Analysis in Software Product Line Engineering (FMSPLE'16) EPTCS206, April 3, 2016, Eindhoven, Netherlands, pp.46-60, 2016

[153] Steiner, E., Masiero, P.: Managing SPL Variabilities in UAV Simulink Models with Pure::variants and Hephaestus, CLEI Electronic Journal 16(1), 16 pages, 2013

[154] Struckmann, W., Wätjen, D.: Mathematik für Informatiker: Grundlagen und Anwendungen, 2. Auflage, Springer-Vieweg, 2016

[155] Sumathi, S., Surekha, P.: Computational Intelligence Paradigms: Theory and Applications using MATLAB, CRC Press, Taylor & Francis Group, 2010

[156] Syberfeldt, A., Rogstörm, J., Geertsen, A.: Simulation-Based Optimization of a Real-World Travelling Salesman Problem Using an Evolutionary Algorithm with a Repair Function, International Journal of Artificial Intelligence and Expert Systems (IJAE), 6(3), pp. 27-39, 2015

[157] Talbi, E.-G.: Metaheuristics: From Design to Implementation, John Wiley & Sons Inc., 2009

[158] The MathWorks: MATLAB R2016a Primer, The MathWorks Inc, Online-Quelle, https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf, Zugriff: 30.08.2016

[159] TheMathWorks: Global Optimization Toolbox, TheMathWorks Inc., 2017, Online-Quelle: https://de.mathworks.com/help/gads/, Zugriff: 30.07.2018

[160] TheMathWorks: MATLAB Central, File Exchange, TheMathWorks Inc., Online-Quelle: https://de.mathworks.com/matlabcentral/fileexchange/?term=type%3AFunction, Zugriff: 30.07.2018

[161] TheMathWorks: Statistics and Machine Learning Toolbox, TheMathWorks Inc., Online-Quelle: https://de.mathworks.com/help/stats/industrial-statistics.html?searchHighlight=industrials&s_tid=doc_srchtitle , Zugriff: 30.7.2018

[162] TheMathWorks: MATLAB: The Language of Technical Computing, TheMathWorks Inc., Online-Quelle: https://de.mathworks.com/help/matlab/, Zugriff: 30.07.2018

[163] TheMathWorks: Sensitivity Analysis – Morris Method (simple), Version 1.8 uploaded by Mr on 29.12.2014, Online-Quelle: https://de.mathworks.com/matlabcentral/fileexchange/48883-sensitivity-analysis-morris-method--simple-, Zugriff: 30.07.2018

[164] TheMathWorks: SIMULINK: Simulation and Model-Based Design, TheMathWorks Inc., Online-Quelle: https://de.mathworks.com/help/simulink/, Zugriff: 30.07.2018

[165] TheMathWorks: SIMULINK: User's Guide 2017a, TheMathWorks Inc., Online-Quelle: https://www.mathworks.com/help/pdf_doc/simulink/sl_using.pdf, Zugriff: 30.07.2018

[166] Traoré, M.K., Muzy, A.: Capturing the dual relationship between simulation models and their context, Simulation Modelling Practice and Theory 14, Elsevier Verlag, pp. 126-142, 2006

[167] Uhrmacher, A.M., Weyns, D.: Multi-Agent Systems: Simulation and Applications, CRC Press, Taylor and Francis Group, 2009

[168] Uhrmacher, A.M.: Seven Pitfalls in Modeling and Simulation Research, In: 2012 Winter Simulation Conference, 9.-12. December, Berlin, GER, pp.3483-3494, 2012

[169] Uttig, M., Leageard, B.: Practical Model-Based Testing: A Tools Approach, Elsevier Science & Technology Books, 2006

[170] VDI 3633: Begriffe, Simulation von Logistik-, Materialfluss- und Produktionssystemen, Düsseldorf, 2013

[171] VDI 3633: Blatt 1, Simulation von Logistik-, Materialfluss- und Produktionssysteme: Grundlagen, Düsseldorf, 2013

[172] von Strauch, R.: Variantenanalyse und ihre Anwendung in der Montage bei Serienfertigung, In: Gaede KW., Pressmar D.B., Schneeweiß C., Schuster KP., Seifert O. (eds), Papers of the 8th DGOR Annual Meeting / Vorträge der 8. DGOR Jahrestagung. Proceedings in Operations Research 8 Physica, Heidelberg, pp.207-208, 1979

[173] Wagner, T., Gellrich, A., Schwenke, C., Kablitzsch, K., Schneider, G.: Automated Planing and creation of Simulation Experiments with a Domain Specific Ontology for Semiconductor Manufacturing AMHS, In: 2014 Winter Simulation Conference, 7-10 December, Savannah, GA, USA, pp. 2628-2639, 2014

[174] Wan, H., Xia, J., Zhang, L., She, D., Xiao, Y., Zou, L.: Sensitivity and Interaction Analysis Based on Sobol‘ Method and Ist Application in a Distributed Flood Forecasting Model, Water, vol. 7, p. 2924-2951, 2015

[175] Weicker, K.: Evolutionäre Algorithmen, Springer Fachmedien, 3. Auflage, 2015

[176] Weiland, J., Manhart, P.: A Classification of Modeling Variability in Simulink, In: VaMoS’14, 22-24 January, Sophia Antipolis, France, pp. 7:1-7:8, 2014

[177] Weißleder, S., Lackner, H.: Zwei Ansätze zur automatischen modellbasierten Generierung von Testfällen für variantenreiche Systeme, Softwaretechnik-Trends 33(2), 4 pages, 2013

[178] Wenzel, S., Weiß, M., Collisi-Böhmer, S., Pitsch, H., Rose, O.: Qualitätskriterien für die Simulation in Produktion und Logistik: Planung und Durchführung von Simulationsstudien, Springer-Verlag, 2008

[179] Withey, J.: Investment Analysis of Software Assets for Product Lines, Technical Report, CMU/SEI-96-TR-010, ESC-TR-96-010, Carnegie Mello University, 1996

[180] Wittern, J. E.: Modeling and Selection of Software Service Variants, Diss. Karlsruher Institut für Technologie (KIT), 2014

[181] Yin, P., Shi, J., Q.: Simulation-Based Sensitivity Analysis for Non-ignorable Missing Data, Cornell University Library, arXiv:1501.05788, 18 pages, 2015 Online-Quelle: https://arxiv.org/abs/1501.05788, Zugriff: 30.07.2018

[182] Zander, J., Schieferdecker, I., Mosterman, J.: Model-based Testing for Embedded Systems, CRC Press, Taylor & Francis Group, 2012

[183] Zander-Nowicka, J.: Model-Based Testing of Real-Time Embedded Systems in the Automotive Domain, Diss. TU Berlin, 2008

[184] Zeigler, B. P., Seo, C., Coop, R., Kim, D.: Creating suites of models with system entity structure: global warming example, In: Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium (DEVS 13), Society for Computer Simulation International, San Diego, Ca, USA, Arcle 32, 8 pages, 2013

[185] Zeigler, B.P.: Multifaceted modelling and discrete event simulation, San Diego, CA, Academic Press, 1984

[186] Zeigler, B.P., Hammonds, P.E.: Modeling and Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange, Academic Press, 2007

[187] Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation: Integration Discrete Event and Continuous Complex Dynamik Systems, 2nd Edition, Academic Press, 2000

[188] Zeigler, B.P., Sarjoughian, H.S.: Guide to Modeling and Simulation of Systems of Systems, Springer-Verlag London, 2013

[189] Zimmer, M.: Durchgängiger Simulationsprozess zur Effizienzsteigerung und Reifegraderhöhung von Konzeptbewertungen in der Frühen Phase der Produktentstehung, Diss. Universität Stuttgart, 2015

[190] Zupancic, B.: Modular hierarchical modelling with SIMCOS language, Mathematical computer simulation, vol. 46, pp. 67-76, 1998

Anhang

A. Spezifikation einer SES in der *Natural Language*

Beispielhafte Spezifikation der SES aus Abbildung 13 in Abschnitt 3.2 in der von [188] *eingeführten natural language*.

//Spezifikation des SES-Baumes
From the Prozesskette-Dec **perspective**, Prozesskette **is made of** Generator, FIFO, APK **and** Senke!

From the APK-Dec1 **perspective**, APK **is made of** ALD, LIFO **and** ARS!

From the APK-Dec2 **perspective**, APK **is made of** ALDs, Puffer, Härten, LIFO **and** ARS!

From the ALD-MAsp **perspective**, ALDs **are made of more than one** ALD!

Puffer **can be** FIFO **or** LIFO **in** Puffer-Spec!

Härten **can be** VH **or** IH **in** Härten-Spec!

//Spezifikation der Attribute der jeweiligen Entitäten
Set Generator´**s** adw **to** ##! *//## steht für einen beliebigen Wert*

Set Generator´**s** matType **to** ##!

Set FIFO´**s** kapa **to** ##!
…

B. Beispiel zur Verwendung von SES-Funktionen

Abbildung 64 zeigt die Modellierung zweier Prozesskettenvarianten mittels der baseline SES nach Zeigler [187] und Zeigler und Hammonds [186] sowie die Modellierung der SES unter verwendung der im Abschnitt 3.3 eingeführten Erweiterungen. Der Unterschiede der beiden Prozesskettenvarianten ist ausschließlich auf die Kopplungsrelation *(ALD.KSS(t).Scope.in2)* begrenzt.

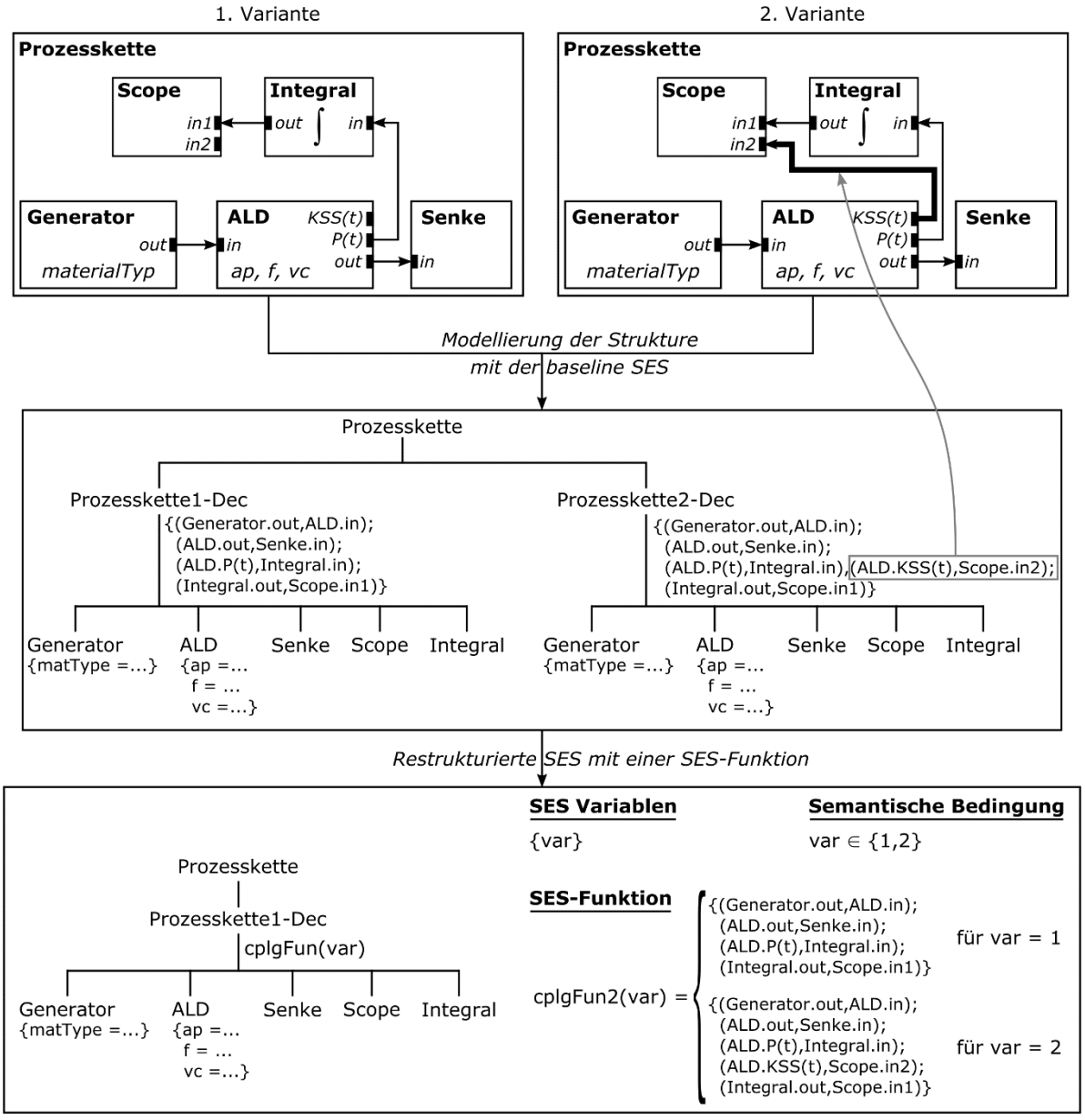


Abbildung 64: Modellierung von zwei Varianten mittels baseline SES und der eingeführten Erweiterungen der SES.

Die Spezifikation der baseline SES ist in der Mitte der Abbiuldung 66 zu sehen. Da der Unterschied auf die Kopplung zu begrenzen ist, muss die Variabilität mittels unterschiedlicher Aspekt-Knoten *Prozesskette1-Dec* und *Prozesskette2-Dec* beschrieben werden. Wie zu erkennen ist, sind beide Teilbäume der genannten Aspekt-Knboten bis auf die Kopplungsrelationen identisch. Beireits eine solche Variabilität kann die SES unnötigerweise aufblähen und im schlimmsten Fall die Übersicht beeinträchtigen. Im

unteren Teil der Abbildung 64 wird die baseline SES mittels der eingeführten Erweiterungen restrukturiert. Dabei kann die Variabilität mit SES-Funktionen spezifiziert und mit der SES-Variablen *var* verknüpft werden. In diesem Fall ist nur ein Aspekt-Knoten notwendig. Beim Pruning wird dann die SES-Funktion anhand der Belegung der SES-Variablen ausgewertet und die entsprechenden Kopplungsrelationen an den Aspekt-Knoten zurückgeliefert. Durch ein solches Vorgehen kann die SES übersichtlicher und minimalistischer spezifiziert werden.

C. Abstrakter pruning Algorithmus

```
1.  def PES = pruning(SES,SESVariables)
2.
3.    //check semantic conditions based on SESVariables
4.    if ~SES.evaluateSemanticConditions
5.      //semantic conditions are false
6.      return PES = [];
7.    else
8.      PES = startPruning([],SES.getRootNode,SESVariables)
9.    end
10.   //check couplings in each aspect node according to entity names and
11.   //modify if necessary
12.   PES = checkAndModifyCouplings(PES);
13.
14. end_def
15.-----------------------------------------------------------------------
--
16. def PES = startPruning(PES,node,SESVariables)
17.
18.   switch node.getType   //determine node type
19.    case 'entity'
20.      //check attributes for SES-Functions and execute SES-Functions
21.      node.checkAndEvaluateSESFunctions(SESVariables);
22.
23.      if entity has more than one aspect
24.        //entity is a variation point
25.        //evaluate selection rules and remove not selected children
26.        node.evaluateSelectionRulesAndRemoveChildren(SESVariables);
27.      end
28.    case 'multiaspect'
29.      //generate children and edges in node based on the SESVariables
30.      //and rename children (axiom valid brother AX4 cf. p.42)
31.      node.generateChildrenAndRenameNodes(SESVariables);
32.    case 'aspect'
33.      //check attributes for SES-Functions and execute SES-Functions
34.      node.checkAndEvaluateSESFunctions(SESVariables);
35.    case 'specialization'
36.      //evaluate selection rules and remove not selected children
37.      node.evaluateSelectionRulesAndRemoveChildren(SESVariables);
38.   end
39.
40.   for each children in node
41.     // start recursion for each children in node
42.     PES = startPruning(PES,node[children],SESVariables);
43.   end
44.
45.   switch node.getType   //determine node type
46.    case 'entity'
47.      if children is a specialization node
48.        //combine names, attributes and edges of node and children of
49.        //specialization (axiom inharitance AX6 cf. p.42)
50.        node.combineEntities(node,node.children.children);
51.      end
52.    case 'specialization'
53.      //specialization nodes are not represented in the PES
54.      return
55.   end
56.
57.   PES.append(node); //append node to PES
58. end_def
```

D. Abstrakter translation Algorithmus

```
1. def ExecutableModel = translation(PES,MB)
2.
3.ExecutableModel=startTranslation(PES.getRootNode, PES.getRootNodeName,…
4.                                  generateEmptyModel(),MB);
5. end_def
6.-------------------------------------------------------------------
7.def ExecutableModel=startTranslation(node,modelPath,ExecutableModel,MB)
8.
9.  switch node.getType  //determine node type
10.   case 'entity'
11.      modelPath = [modelPath, node.name]; //extend modelPath
12.   if node has childen
13.   //inner entity corresponds to coupled model
14.     //generate and add an empty coupled model to ExecutableModel
15.     ExecutableModel.addCoupledModel(modelPath);
16.   else
17.   //leaf entity corresponds to model component of the MB
18.     //get model component based of the attribute mb in node
19.     modelComponent = MB.getModelComponent(node.getMbAttribute);
20.
21.     //parameter setting of model component based on node attributes
22.     //without attibute mb
23.     modelComponent.setParameters(node.getAttributes);
24.
25.     // add model component to ExecutableModel
26.     ExecutableModel.addModelComponent(modelPath,modelComponent);
27.   end
28.  end
29.
30.  for each children in node
31.    //start recursion for each children in node
32.    ExecutableModel = startTranslation(node[children],...
33.                                       modelPath,ExecutableModel,MB);
34.  end
35.
36.  switch node.getType  //determine node type
37.    case 'aspect'
38.      //generate external inputcouplings,external output and internal
39.      //couplings of coupled model based on coupling specification in
40.      //aspect node
41.      ExecutableModel.generateCouplings(modelPath,node.couplings);
42.    end
43. end_def
```

E. Beispiele zum Abschnitt 3.3.5

Abbildung 65 zeigt eine mögliche PES, abgeleitet aus der SES aus Abbildung 17 mit den Belegungen der SES-Variablen

numRep =1, puffer = none, und härten = none

Das ausführbare Simulationsmodell wird durch die Methode *translation* unter Verwendung der PES in Abbildung 65 und der MB aus Abbildung 13 automatisch erzeugt.

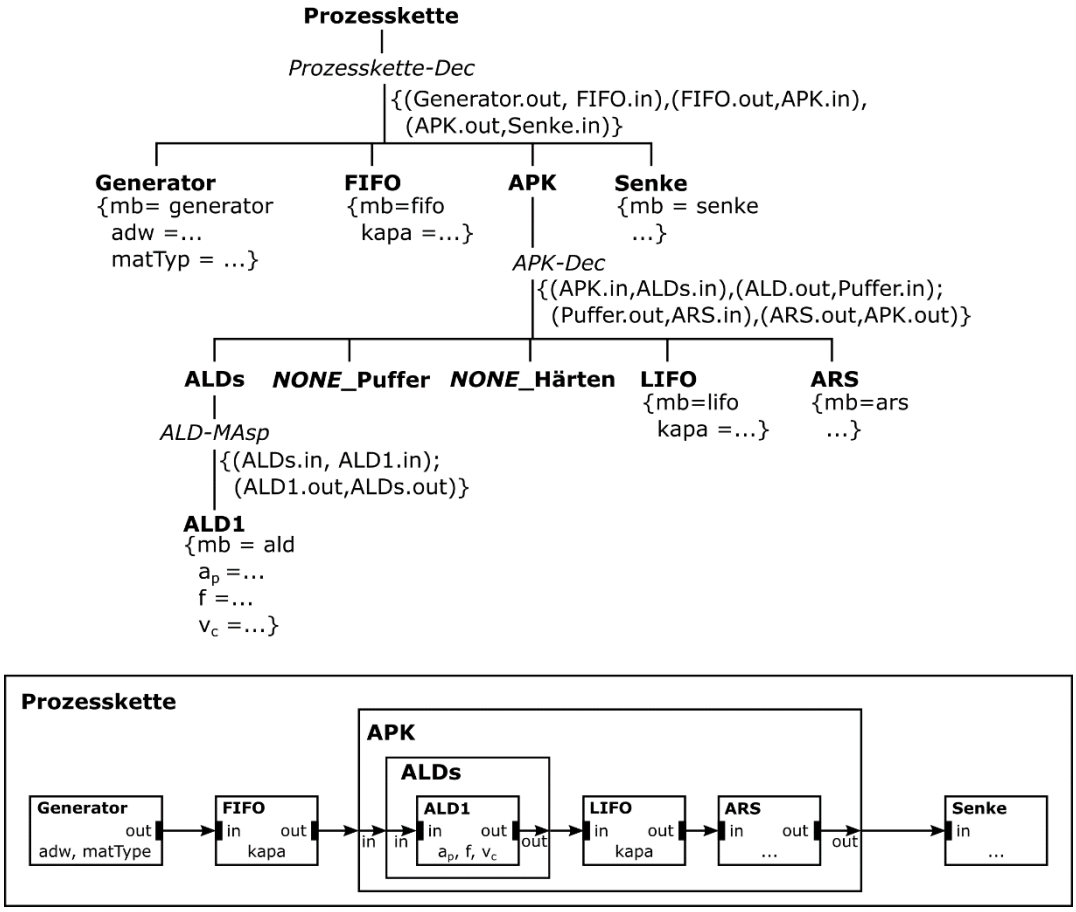


Abbildung 65: PES mit dem zugehörigen Simulationsmodell auf Basis der SES in Abbildung 17 und der MB aus Abbildung 13.

Das Simulationsmodell der Prozesskette in Abbildung 65 entspricht grundlegend der Prozesskette in Abbildung 15 im Unterabschnitt 3.2.2 mit dem Unterschied, dass in Abbildung 65 eine zusätzliche Schicht ALDs hinzugekommen ist. Dieser Sachverhalt ist dem Multi-Aspekt *ALD-MAsp* geschuldet.

Eine weitere PES der SES aus Abbildung 17 und das zugehörige Simulationsmodell sind in Abbildung 66 gezeigt. Diese können mit der Wertebelegung der SES-Variablen

numRep =2, puffer = fifo, und härten = vh

erreicht werden. Die für das Generieren des Simulationsmodells notwendige MB kann der Abbildung 13 entnommen werden.

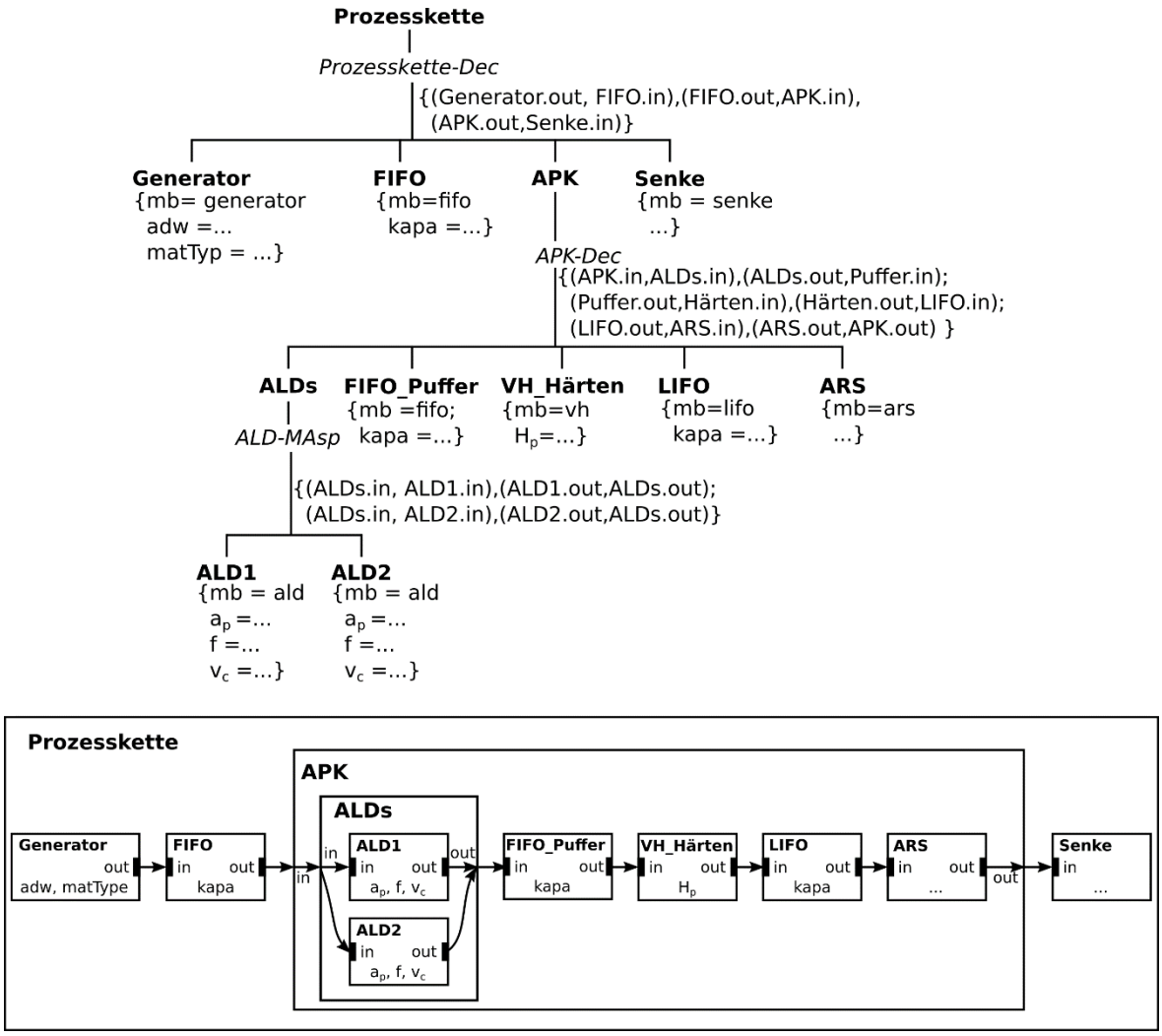


Abbildung 66: PES und das zugehörige Simulationsmodell basierend auf der SES in Abbildung 17 und der MB aus Abbildung 13.

F. Schematische Darstellung einer möglichen Experimentsteuerung in MATLAB/Simulink

```
1.  %% Experiment Control
2.
3.  %% Initialization
4.  % add path
5.  warning off
6.  addpath(genpath('ModelBase'));
7.  addpath(genpath('ExperimentHarness'));
8.  addpath(genpath('Functions'));
9.  addpath(genpath('SESFiles'));
10. addpath(genpath('SESToolbox'));
11.
12. % ses file
13. ses = 'firstExampleSES_07082017.mat';
14. % directory for model base
15. mb = 'ModelBase';
16. % create sesmb object
17. sesmb = SESMB(ses,mb);
18. % create experiment execution unit
19. eu = ExperimentExecutionUnit(@addPathFcn,'parallel');
20.
21. %% ---------------- Experiment Phase Early ----------------
22. % SCREENING: Design of Experiment (DOE) & Elementary Effect (EE)
23.
24. %% SES/MB Configuration (pruning & translation)
25. exp = sesmb.generateExperiments(createSESVars({'EM'},{'sc'},...
26.                                                {'SC'},{'doe','ee'},...
27.                                                {'numRepTM'},{1,2,3},...
28.                                                {'HF'},{1,2,3},...
29.                                                {'TF'},{1,2,3},...
30.                                                {'numRepGM'},{1,2,3}));
31.
32. %% Experiment Execution
33. earlyResults = eu.executeExperiments(exp);
34.
35 .%% Data Collection
36. [rawResultsDOE, rawResultsEE] = separateResults(earlyResults,...
37.                                                  {'SC_doe','SC_ee'});
38. [rawFactorsDOE, Y_DOE] = dataCollectionDOE(rawResultsDOE); % DOE
39. [mueANDstdDev, Y_EE] = dataCollectionEE(rawResultsEE);     % EE
40.
41. %% Analysis
42. factorMatrixDOE = analysisDOE(rawFactorsDOE); % DOE
43. factorMatrixEE = analysisEE(mueANDstdDev);    % EE
44.
45. %% Evaluation
46. [sigFactorsDOE,nonSigFactorsDOE] = evaluation(factorMatrixDOE); % DOE
47. [sigFactorsEE,nonSigFactorsEE] = evaluation(factorMatrixEE);    % EE
```

G. Überschlagsrechnung für die Experimentausführungszeit

Es wird eine Überschlagsrechnung für die Grenzen der maximalen und minimalen Zeitdauer ($t_{E,max}$, $t_{E,min}$) für die Durchführung der komplexen Experimente für alle 81 Prozessketten-varianten für die Experimentmethode *Elementary Effect* und *Sobol* durchgeführt.

$$t_{E,min} = t_{elemE,min} + t_{sob,min}$$

$$t_{E,max} = t_{elemE,max} + t_{sob,max}$$

mit:

$t_{E,min}$ minimale Zeitdauer für die Durchführung komplexer Experimente für alle Varianten.

$t_{E,max}$ maximale Zeitdauer für die Durchführung komplexer Experimente für alle Varianten.

$$t_{elemE,min} = t_{SR,min} * SR_{elemE} * \frac{V}{p}$$

$$t_{elemE,max} = t_{SR,max} * SR_{elemE} * \frac{V}{p}$$

$$t_{sob,min} = t_{SR,min} * SR_{sob} * \frac{V}{p}$$

$$t_{sob,max} = t_{SR,max} * SR_{sob} * \frac{V}{p}$$

mit:

$t_{elemE,min}$ minimale Zeitdauer für die Durchführung komplexer Experimente mit *Elementary Effect* für eine Variante.

$t_{elemE,max}$ minimale Zeitdauer für die Durchführung komplexer Experimente mit *Elementary Effect* für eine Variante.

$t_{sob,min}$ minimale Zeitdauer für die Durchführung komplexer Experimente mit *Sobol* für eine Variante.

$t_{sob,max}$ minimale Zeitdauer für die Durchführung komplexer Experimente mit *Sobol* für eine Variante.

V Anzahl der Varianten.

p Anzahl der Prozessoren.

SR_{elemE} Anzahl der Simulationsläufe für *Elementary Effect.*

SR_{sob} Anzahl der Simulationsläufe für *Sobol.*

Nach Wan et al. [174] berechnen sich für die Experimentmethode *Elementary Effect* die Anzahl durchzuführender Experimente (EE) nach (1) und für die Experimentmethode *Sobol* gemäß (2).

$$SR_{elemE} = N * (k + 1) \qquad (1)$$

$$SR_{sob} = N * (2 * k + 1) \qquad (2)$$

mit:
N Stichprobenumfang
k Anzahl der Faktoren

Für das aktuelle Anwendungsbeispiel ergibt sich:

N = 300, k = 7, V = 81, p = 8

$$t_{elemE,min} = \frac{1.45\ min}{60\frac{min}{h} * 24\frac{h}{T}} * 300 * (7 + 1) * \frac{81}{8} = 24.5\ T$$

$$t_{elemE,max} = \frac{5.4\ min}{60\frac{min}{h} * 24\frac{h}{T}} * 300 * (7 + 1) * \frac{81}{8} = 91.1\ T$$

$$t_{sob,min} = \frac{1.45\ min}{60\frac{min}{h} * 24\frac{h}{T}} * 300 * (2 * 7 + 1) * \frac{81}{8} = 45.9\ T$$

$$t_{sob,max} = \frac{5.4\ min}{60\frac{min}{h} * 24\frac{h}{T}} * 300 * (2 * 7 + 1) * \frac{81}{8} = 170.9\ T$$

Somit ergibt sich

$$t_{E,min} = 24.5T + 45.9T = \mathbf{70.4}\boldsymbol{T}$$

$$t_{E,max} = 91.1T + 170.9T = \mathbf{262}\boldsymbol{T}$$

Thesen

1. Die Methode der Modellbildung und Simulation (M&S) ist in den Ingenieurswissenschaften ein wesentlicher Bestandteil des modernen Entwicklungs- und Planungsprozesses.

2. Die Ansprüche der Kunden nach individuellen technischen Produkten führen zu einer hohen Variantenvielfalt.

3. Die klassischen Methoden der M&S stoßen bei der Betrachtung der Variantenvielfalt an ihre Grenzen.

4. Die modular-hierarchische Modellierung ist eine essentielle Voraussetzung zur Beherrschung der Variantenvielfalt.

5. Der modular-hierarchische Ansatz eignet sich nicht nur für die Modellierung unterschiedlicher Systeme sondern auch für die Spezifikation unterschiedlicher Simulationsexperimente.

6. Das *Experimental Frame*, als Kontextspezifikation, bildet einen zentralen Aspekt zur modular-hierarchischen Spezifikation von Simulationsexperimenten.

7. Simulationsexperimente können in *einfach*, *komplex* und *hochkomplex* kategorisiert werden.

8. Zur Beherrschung der Variantenvielfalt ist ein geordnetes Variantenmanagement, bestehend aus den Phasen *Variantenanalyse*, *Variantenformalisierung*, *Variantenimplementierung* und *Variantengenerierung*, notwendig.

9. Das klassische *System Entity Structure/Model Base Framework* (SES/MB) ist direkt, um Methoden und Komponenten zum Variantenmanagement von Simulationsmodellen und –experimenten, erweiterbar.

10. Die Untersuchung einer hohen Variantenvielfalt erfordert Methoden zur Automatisierung von Simulationsexperimenten.

11. Für das automatisierte Experimentieren ist das SES/MB Framework um eine Experimentsteuerung und eine Experimentausführungseinheit zu erweitern.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kentlich gemacht habe.

_____ _____

Ort, Datum Unterschrift