Spezifikation und Validierung von Zielen für Systems-of-Systems

DISSERTATION
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
durch die Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen

vorgelegt von
Jennifer Brings
geb. in Essen

Betreuer: Prof. Dr. Klaus Pohl
Lehrstuhl: Software Systems Engineering

Essen, 2022

DOI: 10.17185/duepublico/77247
URN: urn:nbn:de:hbz:465-20230405-141520-6

Tag der mündlichen Prüfung: 21. November 2022
Erstgutachter: Prof. Dr. Klaus Pohl
Zweitgutachter: Prof. Dr. Michael Goedicke

Kurzfassung

Ein System-of-Systems ist ein übergeordnetes System, das sich aus unabhängig voneinander funktionierenden Systemen zusammensetzt. Unterschiedliche Zusammensetzungen eines System-of-Systems (SoS-Konfigurationen) ermöglichen die Erfüllung von unterschiedlichen Anforderungen. Daher müssen bei Systems-of-Systems nicht nur die Anforderungen an das System-of-Systems spezifiziert und validiert werden, sondern darüber hinaus muss auch sichergestellt werden, dass für jede SoS-Konfiguration die gewünschten Anforderungen spezifiziert sind. Die zielorientierte Entwicklung erlaubt es, bereits in frühen Phasen Konflikte in Anforderungen zu identifizieren. Hierzu wird die Erfüllbarkeit der an ein System spezifizierten Ziele untersucht. Bei der zielorientierten Entwicklung von Systems-of-Systems muss sichergestellt werden, dass für jede SoS-Konfiguration die gewünschten Ziele spezifiziert und erfüllbar sind.

Die Analyse des Stands der Wissenschaft zeigt, dass es Ansätze aus der Softwareproduktlinienentwicklung gibt, die sich auf die Spezifikation von Zielen für Systems-of-Systems übertragen lassen. Des Weiteren existieren Ansätze zur Analyse solcher Spezifikationen, die automatisiert Fehler identifizieren, die auf Widersprüche zurückzuführen sind. Jedoch gibt es bisher keine Unterstützung für die Prüfung solcher Spezifikationen gegen Stakeholder-Wünsche, bei der die komplexen Beziehungen zwischen Zielen und SoS-Konfigurationen so aufbereitet werden, dass die notwendige manuelle Prüfung erleichtert wird.

Diese Arbeit schlägt einen Ansatz zur Unterstützung der Spezifikation und Validierung von komplexen Zielspezifikationen für Systems-of-Systems vor. Die Lösungsidee basiert auf der Spezifikation von Zielen und SoS-Konfigurationen in getrennten Modellen und der expliziten Verknüpfung dieser Modelle zur Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen. Zur Unterstützung der Validierung wird die automatisierte Generierung von Sichten vorgeschlagen. Die Sichten heben hervor, welche Ziele für ausgewählte SoS-Konfigurationen und welche SoS-Konfigurationen für ausgewählte Ziele spezifiziert wurden.

Zur Evaluation des vorgeschlagenen Ansatzes wird überprüft, ob der Ansatz anwendbar ist und ob er einen Vorteil gegenüber dem aktuellen Stand der Wissenschaft bietet. Hierzu werden ein Fallbeispiel, eine prototypische Implementierung und ein kontrolliertes Experiment herangezogen. Die Anwendung des vorgeschlagenen Ansatzes an einem Fallbeispiel aus der Robotik-Industrie zeigt, dass der Ansatz prinzipiell anwendbar ist. Die prototypische Implementierung des Ansatzes weist nach, dass die Sichten automatisiert erstellt werden können und dass dies in akzeptabler Laufzeit geschieht. Zur Bewertung der Vorteilhaftigkeit wurde ein kontrolliertes Experiment durchgeführt. Durch dieses Experiment wird deutlich, dass die Nutzung der Sichten beim Review Vorteile gegenüber der Nutzung der SoS-Zielspezifikation bietet.

Abstract

A system-of-systems is a high-level system composed of independently functioning systems. Different compositions of a system-of-systems enable the fulfillment of different requirements. Therefore, in the case of systems-of-systems, not only must the requirements for the system-of-systems be specified and validated, but it must also be ensured that the desired requirements are specified for each composition. Goal-oriented development allows conflicts in requirements to be identified in early phases. For this the fulfillment of the goals specified for a system is analyzed. For the goal-oriented development of systems-of-systems it must be guaranteed that for each composition, the desired goals are specified and that these are fulfillable.

An analysis of the state-of-the-art shows there are approaches from software product line engineering, which can be applied to the specification of goals for systems-of-systems. Furthermore, there are approaches for analyzing such specifications that automatically identify errors due to inconsistencies. However, there is currently no support for checking such specifications against stakeholder wishes, in which the complex relationships between goals and compositions are presented in a way that facilitates the necessary manual validation.

This work proposes an approach for supporting the specification and validation of complex goal specifications for systems-of-systems. The solution idea is based on specifying goals and compositions in separate models and explicitly linking these models to specify the relationships between goals and compositions. Automated generation of views is proposed to support validation. The views highlight which goals have been specified for selected compositions and which compositions have been specified for selected goals.

To evaluate the proposed approach, it is examined whether the approach is applicable and whether it offers an advantage over the current state of the art. For this purpose, a case study, a prototype implementation, and a controlled experiment are used. The application of the proposed approach to a case study from the robotics industry shows the approach is applicable. The prototypical implementation of the approach shows the views can be created automatically and that this is done in acceptable time. A controlled experiment was conducted to evaluate the benefits. The experiment shows using the views in the review offers advantages over using the SoS goal specification.

Danksagung

An dieser Stelle möchte ich all den Menschen danken, die mich in den letzten Jahren begleitet und unterstützt und so zum erfolgreichen Abschluss meiner Promotion beigetragen haben.

Ich danke Prof. Dr. Klaus Pohl für die Betreuung der Arbeit und seine hilfreichen Anmerkungen. Für die Übernahme des Zweitgutachtens bedanke ich mich bei Prof. Dr. Michael Goedicke. Außerdem möchte ich Prof. Dr. Stefan Schneegaß für die Übernahme des Vorsitzes der Prüfungskommission danken.

Mein besonderer und ausdrücklicher Dank gilt Prof. Dr. Marian Daun. Seine kontinuierliche inhaltliche Betreuung hat es mir ermöglicht meine wissenschaftlichen Fähigkeiten stetig zu verbessern und an einer Vielzahl von Veröffentlichungen mitzuwirken. Seine jahrelange moralische Unterstützung und wertvolle Freundschaft haben mich ermutigt auch in schwierigen Zeiten nicht aufzugeben und meine Ziele zu verfolgen. Danken möchte ich außerdem Prof. Dr. Bastian Tenbergen, der, zusammen mit Prof. Dr. Marian Daun, die ersten Jahre meiner wissenschaftlichen Karriere sehr geprägt hat. Die vielen gemeinsamen Projekte, an denen sie mich teilhaben ließen, haben mir die Möglichkeit gegeben mich stetig weiterzuentwickeln. Dank gilt auch Prof. Dr. Thorsten Weyer, nicht nur für die vielen hilfreichen Ratschläge, sondern auch für die gute Anleitung in den ersten Jahren meiner Promotionszeit.

Ein außerordentliches Dankeschön geht an meine ehemaligen Kolleginnen und Kollegen. Nicht nur ist diese Arbeit in vielen fachlichen Diskussionen gereift, sondern auch die kollegiale Unterstützung hat mich immer wieder aufgebaut und darin bestärkt mein Vorhaben voranzubringen. Ich danke Torsten Bandyszak und Vanessa Stricker für die vielen wertvollen Tipps und ihre Bereitschaft Teile dieser Arbeit intensiv zu lesen und zu kommentieren. Dank gilt auch Julian Bellendorf, Philipp Bohn, Johanna Buchner, Felix Föcker, Heike Göris, Christian Heinz, Kevin Keller, Birgit Kremer, Andrea Salmon, Selda Saritas und Theresa Wettig.

Ich danke herzlichst meinen ehemaligen studentischen Mitarbeiterinnen Patricia Aluko Obe, Katharina Böse, Lisa Krajinski, Alexandra Pompalla und Viktoria Stenkova. Ihr außerordentlich hoher Einsatz und ihre herausragende Unterstützung sind oft über das Erwartbare hinausgegangen und haben mir Erfolge ermöglicht, die so sonst nicht möglich gewesen wären.

Diese Arbeit entstand im Rahmen des BMBF-Forschungsprojekts CrESt (Collaborative Embedded Systems). Ohne die vielen fachlichen Diskussionen, der intensive Austausch und die partnerübergreifende gute Zusammenarbeit wäre sie kaum möglich gewesen. Hierzu bedanke ich mich bei allen Projektbeteiligten. Besonderer Dank gilt Stefanie Wolf und Birthe Böhm von Siemens und Elham Mirzaei, Martin Neumann und Jan Stefan Zernickel von Insystems, deren Fachwissen zur Problemdefinition und Evaluation dieser Arbeit unabdingbar war.

Nicht zuletzt danke ich meinen Eltern für ihr fortwährendes Verständnis und ihre anhaltende Unterstützung, die mir den Rücken freigehalten haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

GRL Goal-oriented Requirement Language

ILP Integer Linear Programming

SD Strategic Dependency

SMT Satisfiability Modulo Theories

SR Strategic Rationale

Teil I

Einführung

1 Kapitel
Einleitung

Die Spezifikation und Validierung von Anforderungen leistet einen wesentlichen Beitrag zur Entwicklung von Systemen, die den Wünschen der Stakeholder entsprechen. Bei der zielorientierten Entwicklung von Systems-of-Systems muss sichergestellt werden, dass für jede SoS-Konfiguration die gewünschten Ziele spezifiziert und erfüllbar sind.

Diese Arbeit befasst sich mit der zielorientierten Spezifikation und Validierung von Anforderungen für Systems-of-Systems – also eines Systems, das sich aus anderen Systemen zusammensetzt. Zur Einführung in das Thema wird zunächst in Abschnitt 1.1 die Bedeutung der Begriffe Spezifikation und Validierung erläutert. In Abschnitt 1.2 werden die Grundlagen des zielorientierten Requirements Engineering kurz vorgestellt. Abschnitt 1.3 bietet einen Überblick über Systems-of-Systems und erläutert die Bedeutung von SoS-Konfigurationen, d. h. der konkreten Ausprägungen eines System-of-Systems, woraufhin Abschnitt 1.4 näher auf die Zusammenhänge zwischen SoS-Konfigurationen und zielorientiertem Requirements Engineering eingeht. Abschnitt 1.5 stellt die Problemstellung vor, die sich für die zielorientierte Spezifikation und Validierung von Anforderungen für Systems-of-Systems ergibt. Abschnitt 1.6 gibt einen Überblick über den Aufbau dieser Arbeit.

1.1 Spezifikation und Validierung von Anforderungen

Bei der Entwicklung von Systemen ist es wichtig, dass die Anforderungen spezifiziert sind. Die Spezifikation von Anforderungen stellt eine elementare Grundlage für ein gemeinsames Verständnis über das zu entwickelnde System dar und hilft, Wissen persistent zu machen. Darum ist es maßgeblich, dass die Anforderungen richtig spezifiziert sind. Richtig spezifiziert bedeutet, dass die Anforderungen den aktuellen Wünschen der Stakeholder[1] entsprechen. Zur Überprüfung, ob dies der Fall ist, müssen die Anforderungen validiert werden. Validierung befasst sich mit der Beantwortung der Frage, ob das richtige System erstellt wird (Boehm 1984). Das bedeutet, ob das zu entwickelnde System sich so verhält, wie die Stakeholder es

[1] Stakeholder sind gemäß ISO/IEC/IEEE 24765 (2017) Individuen und Organisationen, die einen Bezug zu dem System oder dessen Entwicklung haben.

erwarten. Bei der Systementwicklung wird angenommen, dass die Anforderungen die aktuellen Wünsche der Stakeholder widerspiegeln. Daher werden die Anforderungen üblicherweise als Referenz genutzt, um zu entscheiden, ob das System richtig entwickelt wird. Enthält eine Anforderungsspezifikation Fehler, ist die Gefahr groß, dass das entwickelte System nicht den Stakeholder-Wünschen entspricht. Darum ist es wichtig, Anforderungen zu validieren und so die Gefahr zu reduzieren, dass die Stakeholder-Wünsche unberücksichtigt bleiben.

Die Validierung von Anforderungen bringt das Problem mit sich, dass sie gegen undokumentiertes Wissen erfolgen muss und damit wenig Potenzial für Automatisierung bietet. Einer der gängigsten Ansätze sind manuelle Reviews, die je nach Ansatz ein mehr oder weniger rigides Vorgehen für die Begutachtung von Entwicklungsartefakten vorschreiben. Einer der bekanntesten Ansätze für manuelle Reviews sind Software-Inspektionen nach Fagan (1976).

1.2 Zielorientiertes Requirements Engineering

Zielorientierte Vorgehensweisen haben sich bei der Entwicklung von Software-intensiven Systemen etabliert (Horkoff et al. 2019; Lamsweerde 2001). Diese erlauben es, die Strukturierung von Anforderungen, die Kommunikation zwischen Stakeholdern, die frühzeitigen Identifikation von Konflikten und die Priorisierung von Anforderungen zu erleichtern, und unterstützen bereits in frühen Phasen das Erreichen eines hohen Vollständigkeitsgrades. Das Kernelement zielorientierter Ansätze bilden die Definition, Strukturierung und Relationierung von Zielen. Lamsweerde definiert Ziele wie folgt:

> „A goal is an objective the system under consideration should achieve. Goal formulations thus refer to intended properties to be ensured; they are optative statements as opposed to indicative ones, and bounded by the subject matter."
>
> (Lamsweerde 2001)

Ziele definieren, was ein zu entwickelndes System erreichen soll[2]. Somit geben Ziele Wünsche wieder, welche Eigenschaften das zu entwickelnde System aufweisen soll.

Es existieren diverse Zielmodellierungs- und -analyseansätze. Zielmodellierungsansätze unterstützen die Erhebung, Spezifikation und Validierung von Anforderungen (z. B. (Bresciani et al. 2004; Dardenne et al. 1993; ISO/IEC/IEEE 15288 2015; Yu 1997)). Zu den Hauptvorteilen der Verwendung von Zielen in der Systementwicklung gehören:

- Zielmodelle sind lösungsneutral (Yu 1997). Sie können gut bereits in frühen Entwicklungsphasen eingesetzt werden, um Stakeholder-Wünsche zu dokumentieren (Horkoff und Yu 2016).

[2] Im Rahmen dieser Arbeit wird das Modalverb sollen im Sinne des allgemeinen Sprachgebrauchs genutzt und meint eine Anweisung (vgl. Bedeutung 1a) https://www.duden.de/rechtschreibung/sollen). Im juristischen Sprachgebrauch deutet sollen einen Ermessensspielraum an, der hier nicht gegeben ist.

- Zielmodelle nutzen leicht verständliche Sprachen, daher eigenen sie sich gut, um Stakeholder-Wünsche zu dokumentieren und zu prüfen Nuseibeh et al. (1994)
- Vielen Zielmodellierungssprachen, wie bspw. iStar (Yu 1996) oder Goal-oriented Requirement Language (GRL) (International Telecommunication Union 2018), liegt eine formale Semantik zugrunde. Hierdurch wird die Prüfung von Zielmodellen dahingehend unterstützt, dass Zielanalyseansätze aufbauend auf den spezifizierten Zielen verschiedene Arten der Zielerfüllung analysieren (z. B. (Giorgini et al. 2003 2005; Horkoff und Yu 2013; Sebastiani et al. 2004)) oder Zielkonflikte identifizieren (z. B. (Ali et al. 2013; Elahi und Yu 2011; Jureta et al. 2010)) können.

1.3 System-of-Systems

Diese Vorteile des zielorientierten Requirements Engineering kommen auch bei der Entwicklung von Systems-of-Systems zum Tragen. In der Literatur findet sich eine Vielzahl von Definitionen für den Begriff System-of-Systems (Nielsen et al. 2015). Die Grundidee hinter der Betrachtung von Systems-of-Systems lautet, dass ein übergeordnetes System nicht aus Komponenten, sondern aus anderen Systemen, die prinzipiell unabhängig voneinander funktionieren können, zusammengesetzt wird. Dies spiegelt sich in gängigen Definitionen internationaler Standards wider:

> System of systems (SoS)
>
> 1. system-of-interest (SOI) whose elements are themselves systems
> 2. large system that delivers unique capabilities, formed by integrating independently useful systems
>
> (ISO/IEC/IEEE 24765 2017)

Bei einem System-of-Systems erfolgt der Zusammenschluss verschiedener Einzelsysteme zu einem übergeordneten System, da das System-of-Systems so in die Lage versetzt wird, einen Zweck zu erfüllen, den die einzelnen Bestandteile nicht erfüllen können:

> „A system of systems (SoS) brings together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals“
>
> (ISO/IEC/IEEE 15288 2015)

Darüber hinaus schreiben viele Autoren Systems-of-Systems noch weitere Eigenschaften zu (Nielsen et al. 2015).

- *Autonomie der Bestandteile* beschreibt die Eigenschaft, dass die Bestandteile ihre eigenen Ziele nach ihren eigenen Regeln verfolgen (vgl. bspw. (Maier 1998; Ackoff 1971; Noam 1994; Sage und Cuppan 2001)).

- *Unabhängigkeit* beschreibt die Eigenschaft, dass die Bestandteile auch ohne das System-of-Systems Ziele verfolgen und erfüllen (vgl. bspw. (Jamshidi 2008; Sharawi et al. 2006)).

- *Verteilung* beschreibt die Eigenschaft, dass die Bestandteile räumlich voneinander getrennt sein können, da sie nur in der Lage sein müssen, Informationen auszutauschen und nicht notwendigerweise Masse oder Energie (vgl. bspw.(Eisner et al. 1991; Noam 1994; Kotov 1999; Pei 2000; Sage und Cuppan 2001; Boardman und Sauser 2006)).

- *Evolution* beschreibt die Eigenschaft, das System-of-Systems durch die Aktualisierung oder den Austausch von Bestandteilen zu aktualisieren (vgl. bspw. (Maier 1998; Lukasik 1998; Sage und Cuppan 2001; Chen und Clothier 2003; Abbott 2006; Sharawi et al. 2006)).

- *Dynamische Rekonfiguration* beschreibt die Eigenschaft, dass sich zur Laufzeit die Zusammensetzung des System-of-Systems ändern kann, d. h., dass Bestandteile aus dem System-of-Systems entfernt werden oder hinzukommen können (vgl. bspw. (Pei 2000; Chen und Clothier 2003; Keating et al. 2003; Abbott 2006; Sharawi et al. 2006)).

- *Emergentes Verhalten* beschreibt die Eigenschaft, dass sich das Verhalten des System-of-Systems aus dem Zusammenspiel der Bestandteile ergibt (vgl. bspw. (Eisner et al. 1991; Noam 1994; Kotov 1999; Chen und Clothier 2003; Bar-Yam et al. 2004; Boardman und Sauser 2006; Sharawi et al. 2006)).

- *Wechselbeziehungen* beschreiben die Eigenschaft, dass die Bestandteile voneinander abhängig sind, um gemeinsam das Ziel des System-of-Systems zu erreichen (vgl. bspw. (Ackoff 1971; Eisner et al. 1991; Chen und Clothier 2003; Keating et al. 2003; Bar-Yam et al. 2004)).

- *Interoperabilität* beschreibt die Eigenschaft, dass heterogene Bestandteile miteinander interagieren können (vgl. bspw. (Ackoff 1971; Eisner et al. 1991; Noam 1994; Shenhar 1994; Krygiel 1999; Pei 2000; Sage und Cuppan 2001; Bar-Yam et al. 2004; Boardman und Sauser 2006)).

Für diese Arbeit ist vor allem die Eigenschaft der dynamischen Rekonfiguration zentral. Durch dynamische Rekonfiguration ergeben sich unterschiedliche Ausprägungen eines System-of Systems, die gewisse Ähnlichkeiten besitzen, d. h. teilweise gleiche Bestandteile aufweisen. Dies bietet Vorteile für die Systementwicklung, die vor allem im Produktlinien-Engineering berücksichtigt werden. Hierbei wird zwischen festen und variablen Systembestandteilen unterschieden. D. h., Systemzusammensetzungen weisen zum einen identische Bestandteile, zum anderen aber auch unterschiedliche Bestandteile auf. Für solche Systeme existieren

somit unterschiedliche mögliche Systemzusammensetzungen, die sich durch die einzelnen Systembestandteile definieren.

Neben Systems-of-Systems kennt die Literatur weitere Systemtypen, die sich aus verschiedenen Bestandteilen zusammensetzen.

Diese Bestandteile werden je nach Systembetrachtung Komponenten (Crnkovic 2001), Features (Apel und Kästner 2009), Subsysteme (Keating et al. 2003), Services (Turner et al. 2003) o. ä. genannt und zu einem System kombiniert.

Eine häufig verwendete Definition zur Unterscheidung von System-of-Systems und komponentenbasierten Systemen liefert Maier (1998):

> A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties:
>
> - Operational Independence of the Components: If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. That is,the components fulfill customer-operator purposes on their own.
> - Managerial Independence of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems
>
> (Maier 1998)

Während einerseits hiermit die Abgrenzung von Systems-of-Systems von komponentenbasierten Systemen gezeigt wird, wird jedoch andererseits auch die Nähe zu Service-basierten Systemen deutlich, die diese Eigenschaften teilen. Service-basierte Systeme bestehen aus einer Orchestration eigenständiger Services, die im Zusammenspiel einen größeren Service anbieten (Valipour et al. 2009). Daher können auch Service-basierte Systeme als Systems-of-Systems angesehen werden (Lewis et al. 2011). Häufig wird jedoch bei Service-basierten Systemen hauptsächlich die Software betrachtet, während bei Systems-of-Systems oftmals von Systemen mit einer engen Verschmelzung von Hard- und Software ausgegangen wird (Lewis et al. 2011). Diese Unterscheidung ist für den weiteren Verlauf dieser Arbeit allerdings nicht relevant, so kann die Lösungsidee auch auf Service-basierte Systeme angewandt werden.

Da sich Systems-of-Systems aus Systemen zusammensetzen, lassen sich verschiedene Ausprägungen unterscheiden, je nachdem, welche Systeme Bestandteil der jeweiligen Ausprägung ist. Diese Ausprägungen werden als System-of-Systems-Konfiguration bezeichnet.

Definition: System-of-Systems-Konfiguration (SoS-Konfiguration)

Eine SoS-Konfiguration definiert eine mögliche konkrete Ausprägung eines System-of-Systems anhand ihrer Bestandteile aus Einzelsystemen.

Ein Beispiel für ein System-of-Systems ist eine dynamische Transportroboterflotte, die sich aus einer variablen Anzahl von unterschiedlichen Transportrobotern unterschiedlicher Art zusammensetzen kann. Beispielsweise kann eine Konfiguration aus drei Standardrobotern und zwei Schwerlastrobotern bestehen oder eine andere Konfiguration aus fünf Standardrobotern.

Abbildung 1.1 illustriert die Zusammenhänge. Für jedes System-of-Systems gibt es bestimmte SoS-Bestandteile, die Teil dieses System-of-Systems sein können (siehe Abbildung 1.1a). In jeder SoS-Konfiguration sind bestimmte SoS-Bestandteile vorhanden. Abbildung 1.1b) zeigt zwei Beispiele für mögliche SoS-Konfigurationen. SoS-Konfigurationen unterscheiden sich untereinander durch unterschiedliche Bestandteile.

Jede Änderung an der Zusammensetzung des System-of-Systems führt zu einer unterschiedlichen SoS-Konfiguration.

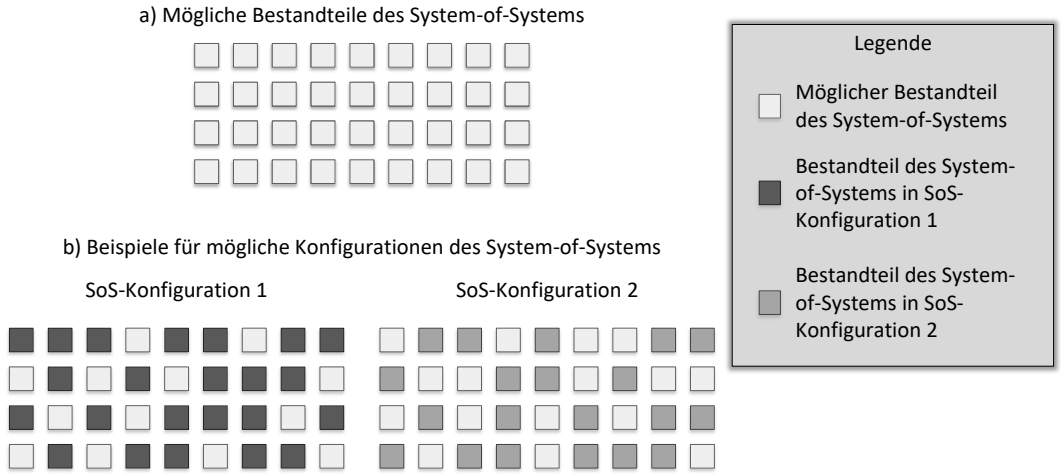


Abbildung 1.1: SoS-Konfigurationen

1.4 Zielorientierung für Systems-of-Systems

Für die Definition von Zielen für Systems-of-Systems bedeutet dies, dass sowohl für die Bestandteile als auch für das System-of-Systems Ziele spezifiziert werden müssen[3]. Bei der Spezifikation der Ziele für das System-of-Systems müssen zudem die verschiedenen SoS-Konfigurationen berücksichtigt werden. So soll ein System in einer SoS-Konfiguration andere Ziele erfüllen als das System in einer anderen SoS-Konfiguration. Eine Transportroboterflotte, der mindestens ein Transportroboter für den Transport von schweren Lasten angehört, kann das Ziel *Schwere Lasten transportieren* erfüllen, während alle Transportroboterflotten, die keinen geeigneten Transportroboter besitzen, dieses Ziel nicht erfüllen können. Auch die Ziele

[3] Diese Arbeit befasst sich mit der Spezifikation der Ziele für das System-of-Systems, nicht mit der Spezifikation der Ziele der Bestandteile des System-of-Systems.

können folglich dahingehend differenziert werden, ob sie in allen oder nur in bestimmten SoS-Konfigurationen erreicht werden sollen. Teilweise ist es auch erforderlich sicherzustellen, dass manche Ziele in einer bestimmten SoS-Konfiguration nicht erreicht werden dürfen. Beispielsweise muss bei einer Transportroboterflotte sichergestellt werden, dass das Ziel *Flüssigkeit transportieren* nicht erreicht wird, wenn der Transportroboterflotte kein geeigneter Transportroboter angehört. Diese Zusammenhänge sind in Abbildung 1.2 illustriert. 1.2a) zeigt alle möglichen Bestandteile sowie die möglichen erreichbaren Ziele des System-of-Systems, die sich aus diesen Bestandteilen zusammensetzen. In 1.2b) sind zwei mögliche SoS-Konfiguration und die jeweils erreichbaren Ziele illustriert.

Das bedeutet, dass für System-of-Systems für jede mögliche SoS-Konfiguration geprüft werden muss, ob

a) gewünschte Ziele korrekt spezifiziert sind.

b) nicht gewünschte Ziele korrekt nicht spezifiziert sind.

Gewünscht bedeutet, dass ein Ziel den Wünschen der Stakeholder entspricht. Nicht gewünscht bedeutet, dass ein Ziel nicht den Wünschen der Stakeholder entspricht. Spezifiziert bedeutet, dass ein Ziel Bestandteil der Zielspezifikation ist. Nicht spezifiziert bedeutet, dass ein Ziel nicht Bestandteil der Zielspezifikation ist. Die Spezifikation eines Ziels deutet an, dass dieses Ziel gewünscht ist. Jedoch kann sich die Situation einstellen, dass gewünschte Ziele nicht spezifiziert wurden oder Ziele spezifiziert wurden, die nicht gewünscht sind.

Neben der Prüfung, ob gewünschte Ziele spezifiziert und nicht gewünscht Ziele nicht spezifiziert sind, ist bei Zielspezifikationen ebenfalls von Bedeutung, ob die spezifizierten und gewünschten Ziele erfüllbar sind.

Erfüllbar bedeutet, dass es keine Widersprüche zwischen spezifizierten Zielen gibt, die die Erfüllung des Ziels verhindern. Nicht erfüllbar bedeutet hier, dass Widersprüche zwischen den spezifizierten Zielen vorliegen, die die Erfüllung des Ziels verhindern.

Widersprüche zwischen Zielen behindern die richtige Implementierung des spezifizierten Systems. Daher ist es wichtig, diese frühzeitig zu erkennen. Außerdem können Widersprüche zwischen Zielen auf Widersprüche zwischen Stakeholder-Wünschen oder eine fehlerhafte Spezifikation von Stakeholder-Wünschen hindeuten.

Nicht betrachtet wird, ob Ziele tatsächlich erfüllt oder nicht erfüllt werden. Erfüllt ist ein Ziel, wenn das System tatsächlich so implementiert wurde, dass das Ziel erreicht wird. Nicht erfüllt ist ein Ziel, wenn das System so implementiert wurde, dass das Ziel nicht erreicht wird. Dies ist auf Basis der Spezifikation der Ziele an ein System nicht überprüfbar, sondern erst nach der Implementierung des Systems.

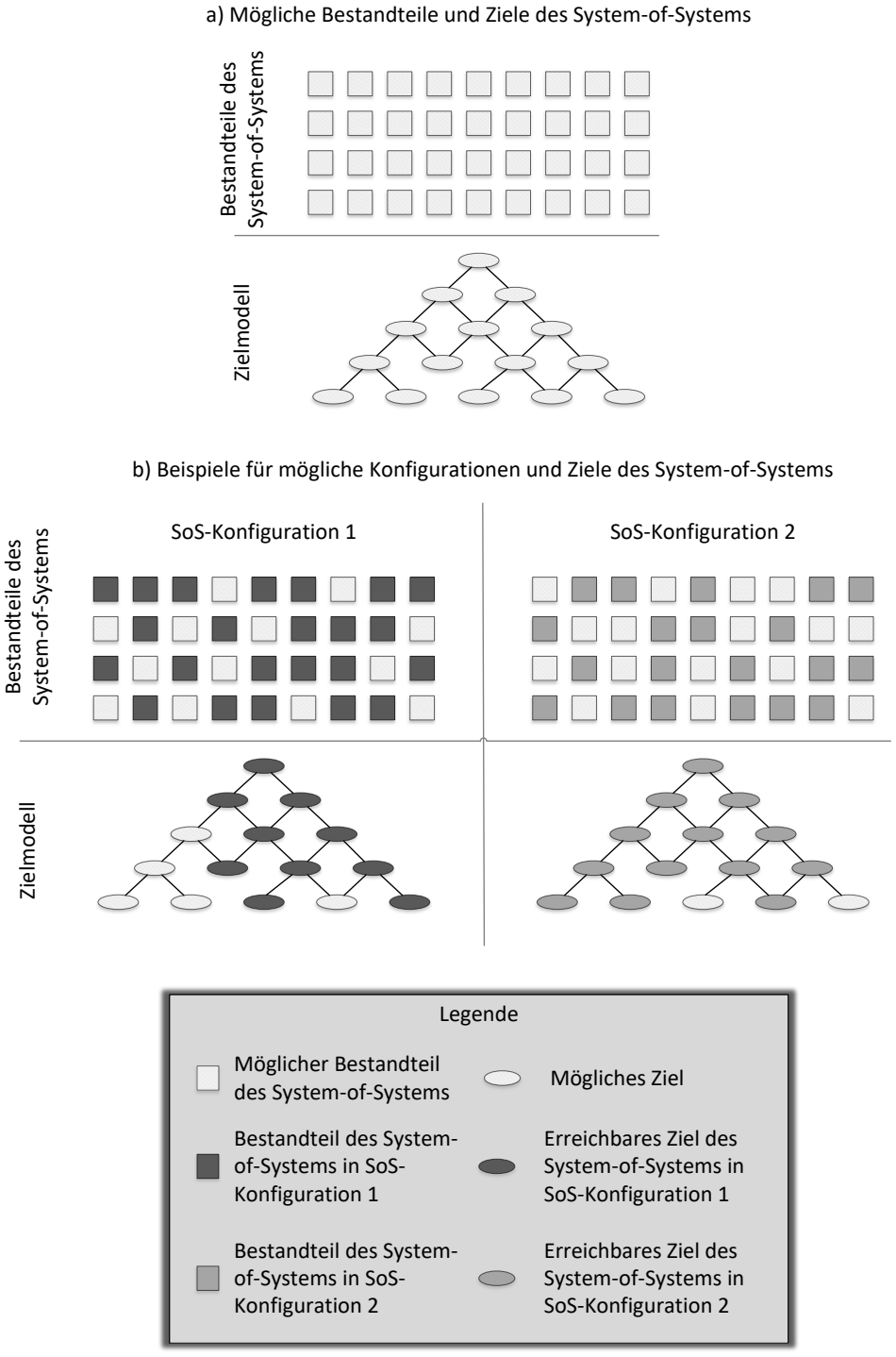


Abbildung 1.2: Zusammenhang zwischen SoS-Konfigurationen und Zielerreichung

1.5 Problemstellung

Bei der zielorientierten Entwicklung von Systems-of-Systems müssen somit die gewünschten Ziele für das zu entwickelnde System-of-Systems spezifiziert werden. Zudem müssen die Ziele auch für alle SoS-Konfigurationen validiert werden, d. h., es muss überprüft werden, ob diese auch den aktuellen Stakeholder-Wünschen entsprechen. Da dies aufgrund der Vielzahl der möglichen SoS-Konfigurationen eine sehr komplexe und kaum überschaubare Tätigkeit ist, benötigt der Entwickler hierbei automatisierte Unterstützung. Diese Arbeit betrachtet daher das Problem:

Bei der zielorientierten Entwicklung von Systems-of-Systems wird Unterstützung benötigt, um Ziele so zu spezifizieren und die Spezifikation so zu validieren, dass sichergestellt wird, dass für jede SoS-Konfiguration die Ziele spezifiziert werden, die von den Stakeholdern für diese SoS-Konfiguration gewünscht sind, und diese Ziele müssen auch in der jeweiligen SoS-Konfiguration erfüllbar sein.

Dies bedeutet:

1. Für jede SoS-Konfiguration müssen die Ziele gemäß den Stakeholder-Wünschen spezifiziert werden. Demzufolge muss für jedes Ziel spezifiziert werden, in welchen SoS-Konfigurationen es erfüllt werden soll und in welchen SoS-Konfigurationen es nicht erfüllt werden soll. Beispielsweise soll das Ziel *Schwere Lasten transportieren* in SoS-Konfigurationen erfüllt werden, die mindestens einen Transportroboter für schwere Lasten besitzen.

2. Für jede mögliche SoS-Konfiguration muss geprüft werden können, ob die Ziele gemäß den Stakeholder-Wünschen spezifiziert sind. Das bedeutet, dass für jedes Ziel geprüft werden muss, ob es für alle SoS-Konfigurationen spezifiziert wurde, in denen es laut Stakeholder-Wünschen erfüllt werden soll, und dass es nicht in SoS-Konfigurationen spezifiziert wurde, in denen es laut Stakeholder-Wünschen nicht erfüllt werden soll. Beispielsweise muss geprüft werden, ob das Ziel *Schwere Lasten transportieren* wirklich nur für SoS-Konfigurationen spezifiziert wurde, die mindestens einen Transportroboter für schwere Lasten besitzen.

3. Bei der Analyse der Zielerfüllung und Konflikte müssen die verschiedenen SoS-Konfigurationen berücksichtigt werden. So kann es sein, dass ein Ziel, das zur Zielerfüllung eines anderen Ziels benötigt wird, nicht in allen SoS-Konfigurationen spezifiziert wurde, in denen das andere Ziel spezifiziert wurde. Beispielsweise könnte das Ziel *Schwere Lasten transportieren* Unterziel von *Besondere Lasten transportieren* sein. Wenn für eine SoS-Konfiguration das Ziel *Besondere Lasten transportieren* spezifiziert wurde, aber alle Unterziele in dieser SoS-Konfiguration nicht erfüllbar sind, ist auch das spezifizierte Ziel *Besondere Lasten transportieren* nicht erfüllbar. Darüber hinaus müssen Konflikte unter Berücksichtigung der jeweiligen SoS-Konfigurationen geprüft werden. Dies ist

erforderlich, da Konflikte zwischen Zielen nur dann problematisch sind, wenn diese Ziele in der gleichen SoS-Konfiguration erfüllbar sein sollen. Beispielsweise wäre ein Konflikt zwischen den Zielen *Schwere Lasten transportieren* und *Verschleiß gering halten* irrelevant, wenn es keine SoS-Konfiguration gibt, für die beide Ziele spezifiziert sind.

Aufgrund der Vielzahl der möglichen SoS-Konfigurationen und deren Gemeinsamkeiten und Unterschiede besteht hierin eine umfangreiche und komplexe Aufgabe, die ohne technische Unterstützung kaum zu bewältigen ist.

Zielsetzung der Arbeit

a) Unterstützung bei der Spezifikation, Validierung und Erfüllbarkeitsprüfung von Zielen für Systems-of-Systems.

b) Unterstützung bei der Berücksichtigung unterschiedlicher SoS-Konfigurationen des System-of-Systems.

Dabei ist zum einen zu beachten, welche SoS-Konfigurationen das System-of-Systems aufweisen soll. Zum anderen müssen die Ziele für jede SoS-Konfiguration spezifiziert, validiert und deren Erfüllbarkeit geprüft werden.

1.6 Aufbau der Arbeit

Diese Arbeit gliedert sich in vier Teile:

Teil I präsentiert die das Thema der Arbeit betreffenden Grundlagen sowie den Stand der Wissenschaft.

- Kapitel 2 führt die Grundlagen ein, die im Rahmen der Lösungsidee genutzt werden, um die Problemstellung zu bearbeiten.
- Kapitel 3 stellt den Stand der Wissenschaft vor und zeigt auf, welche Bereiche der Zielsetzung bereits durch existierende Ansätze abgedeckt sind und welche Lücke noch besteht, die in dieser Arbeit adressiert wird.

Teil II stellt den vorgeschlagenen Lösungsansatz im Detail vor.

- Kapitel 4 präsentiert die Lösungsidee und zeigt, wie diese auf dem Stand der Wissenschaft aufbaut und welche Beiträge die Arbeit liefert.
- Kapitel 5 behandelt, wie Ziele und SoS-Konfigurationen spezifiziert werden, um darauf aufbauend die Validierung und Erfüllbarkeitsanalyse zu unterstützen.
- Kapitel 6 führt aus, wie Ziel- und SoS-Konfigurationssicht erstellt werden, um die Validierung zu unterstützten.

- Kapitel 7 stellt dar, wie Ziel- und SoS-Modelle automatisiert geprüft werden, um ungültige Sichten zu identifizieren bzw. deren Erstellung zu verhindern.

Teil III evaluiert den vorgestellten Lösungsansatz

- Kapitel 8 bietet einen Überblick über das Evaluationskonzept, mit dem der vorgestellte Lösungsansatz bewertet wird.
- Kapitel 9 stellt die Evaluation der Anwendbarkeit des Lösungsansatzes vor. Dies wird durch eine Anwendung des Lösungsansatzes auf ein Fallbeispiel sowie durch eine prototypische Implementierung des Lösungsansatzes gezeigt.
- Kapitel 10 befasst sich mit der Evaluation der Vorteilhaftigkeit des Lösungsansatzes. Zu diesem Zweck wird ein kontrolliertes Experiment durchgeführt.

Teil IV beschließt die Arbeit mit Kapitel 11.

2 Kapitel
Grundlagen

Zur Spezifikation der Ziele des System-of-Systems verwendet diese Arbeit die Goal-oriented Requirement Language (GRL). Die Spezifikation der SoS-Konfigurationen erfolgt anhand eines SoS-Modells, dem kardinalitätsbasierte Feature-Modelle zugrunde liegen. Die Validierung der Zielspezifikation eines System-of-Systems wird durch die Generierung von Sichten unterstützt.

Dieses Kapitel stellt die Grundlagen vor, die im Rahmen der Lösungsidee herangezogen werden, um die Problemstellung zu adressieren. Abschnitt 2.1 beinhaltet zunächst einen kurzen Überblick über die Zielmodellierung. In Abschnitt 2.2 werden die Zielmodellierungssprache GRL, die zur Spezifikation der Ziele des System-of-Systems verwendet wird, und in Abschnitt 2.3 kardinalitätsbasierte Feature-Modelle, die der Spezifikation der SoS-Konfigurationen dienen, vorgestellt. In Abschnitt 2.4 wird ein Überblick über Validierungsansätze für Anforderungen gegeben. Außerdem wird in Abschnitt 2.5 das Grundprinzip der Sichtenbildung erläutert, das im Rahmen der Lösungsidee zur Unterstützung der Validierung Anwendung findet.

2.1 Zielmodellierung

Es existieren diverse Zielmodellierungansätze. Zielmodellierungsansätze unterstützen die Erhebung, Spezifikation und Validierung von Anforderungen. Zu den am weitesten verbreiteten Ansätzen gehören KAOS (Dardenne et al. 1993; Lamsweerde 2009)), iStar (Yu 1997), Tropos (Bresciani et al. 2004), NFR (Mylopoulos et al. 1992) und GRL (International Telecommunication Union 2018).

Zielmodelle besitzen üblicherweise eine Baum- oder Graphen-Struktur. Die Knoten stellen Ziele dar und die Kanten Beziehungen zwischen Zielen. Die Beziehungen beschreiben, inwiefern die Erfüllung eines Ziels das andere Ziel beeinflusst. Viele Zielmodellierungssprachen unterscheiden verschiedene Arten von Zielen, wie bspw. Softgoals in KAOS (Dardenne et al. 1993; Lamsweerde 2009), iStar (Yu 1997), Tropos (Bresciani et al. 2004) und GRL (International Telecommunication Union 2018), und manche stellen auch Modellierungskonstrukte für Dinge zur Verfügung, die über reine Ziele hinausgehen, aber Einfluss auf die

Zielerfüllung nehmen können, wie bspw. Tasks oder Ressourcen in iStar (Yu 1997), Tropos (Bresciani et al. 2004) und GRL (International Telecommunication Union 2018).

Es existieren verschiedene Arten von Beziehungen zwischen Zielen, die in Zielmodellen dargestellt werden können. Häufig handelt es sich um Dekompositionsbeziehungen, die es erlauben, eine Hierarchisierung von Zielen vorzunehmen, wodurch die Definition von Ober- und Unterzielen möglich wird (Horkoff et al. 2019). Oberziele gelten als verwirklicht, wenn je nach Art der Dekompositionsbeziehung mindestens eins oder alle Unterziele erfüllt sind. Des Weiteren bieten viele Zielmodellierungssprachen auch die Möglichkeit, positive und negative Einflüsse bzgl. der Zielerfüllung zwischen den Zielen darzustellen. Je nach Zielmodellierungssprache gelten mehr oder weniger restriktive Einschränkungen, welche Arten von Zielen mit welchen Arten von Beziehungen verbunden werden dürfen. So schreibt beispielsweise iStar (Yu 1997) den Einsatz von Means-End-Links zur Verbindung eines Task mit einem Goal vor, während GRL (International Telecommunication Union 2018) hier auch den Einsatz von Dekompositions-Links gewährt.

Manche Zielmodellierungssprachen erlauben es, Ziele verschiedenen Entitäten – oft Akteure oder Agenten genannt – zuzuordnen. So wird es möglich, zusätzlich Abhängigkeitsbeziehungen zwischen Akteuren zu spezifizieren. Bei KAOS (Dardenne et al. 1993; Lamsweerde 2009)) wird einem Agenten durch ein Responsibility-Assignment die Verantwortung für die Erfüllung eines Ziels übertragen. Bei iStar (Yu 1997) werden die Ziele eines Akteurs innerhalb des Akteurssymbols dargestellt. Diese Art von Modell wird als Strategic Rationale (SR)-Modell bezeichnet. In Strategic Dependency (SD)-Modellen werden die Akteure ohne ihre Ziele dargestellt, um besondere Aufmerksamkeit auf die Abhängigkeiten zwischen den Akteuren zu lenken.

2.2 Goal-oriented Requirements Language

Die GRL ist eine Modellierungssprache für Ziele (International Telecommunication Union 2018). Neben Regeln zur Erstellung von Zielmodellen existieren auch Regeln zur Analyse von Zielmodellen, mit denen sich die Erfüllbarkeit von Zielen prüfen lässt.

2.2.1 Modellierung

Die GRL erlaubt es, Ziele in einem hierarchischen Modell darzustellen und sie auf diese Weise zu verfeinern. Außerdem können weitere Arten von Beeinflussungen zwischen Zielen ausgedrückt werden.

Ein GRL-Zielmodell besteht gemäß International Telecommunication Union (2018) aus Akteuren, intentionalen Elementen und Verbindungen. Ein Zielmodell GM ist definiert als:

$$GM = \langle A, E, L, \alpha \rangle$$

wobei A die Menge der Akteure, E die Menge der intentionalen Elemente und L die Menge der Verbindungen ist. Akteure stellen Systeme, Subsysteme oder Stakeholder dar, denen intentionale Elemente zugewiesen werden können. Hierzu wird die Relation $\alpha \subseteq A \times E$ definiert, wobei jedes intentionale Element höchstens einem Akteur zugeordnet sein kann. Es gilt:

$$\nexists e \in E \wedge a_1, a_2 \in A \mid (a_1, e), (a_2, e) \in \alpha$$

Um die Elemente eines Tupels $\langle a, e \rangle \in \alpha$ zu identifizieren, wird definiert, dass $\alpha(a) = e \mid \langle a, e \rangle \in \alpha$ und $\alpha(e) = a \mid \langle a, e \rangle \in \alpha$.

Ein intentionales Element ist entweder ein Ziel, ein Softgoal, eine Task, eine Ressource oder ein Belief. Das heißt,

$$E = G \cup S \cup T \cup R \cup B$$

wobei G die Menge der Ziele, S die Menge der Softgoals, T die Menge der Tasks, R die Menge der Ressourcen und B die Menge der Beliefs ist. Ziele stellen Zustände dar, die erreicht werden sollen. Softgoals stellen Ziele dar, deren Erfüllung nicht objektiv beurteilt werden kann. Tasks stehen für eine Möglichkeit, etwas zu tun. Ressourcen sind Objekte und Beliefs werden zur Dokumentation von Entwurfsentscheidungen verwendet.

In einem GRL-Zielmodell kann es drei verschiedene Arten von Verbindungen geben: Dekompositionen, Contributions und Dependencies. Das heißt,

$$L = Decomp \cup Con \cup Depend$$

wobei $Decomp$ die Menge der Dekompositonen, Con die Menge der Contributions und $Depend$ die Menge der Dependencies ist.

Bei den Dekompositionen werden drei verschiedene Arten unterschieden: AND-Dekompositionen, XOR-Dekompositionen und IOR-Dekompositionen. Das heißt,

$$Decomp = Decomp^{AND} \cup Decomp^{IOR} \cup Decomp^{XOR}$$

wobei $Decomp^{AND}$ die Menge der AND-Dekompositionen, $Decomp^{IOR}$ die Menge der IOR-Dekompositionen und $Decomp^{XOR}$ die Menge der XOR-Dekompositionen ist. Dabei gilt, dass jede Dekompositon nur einer Art zuzuordnen ist. Das heißt:

$$Decomp^{AND} \cap Decomp^{IOR} = Decomp^{AND} \cap Decomp^{XOR} = Decomp^{IOR} \cap Decomp^{XOR} = \emptyset$$

Eine Dekompositon relationiert zu einem intentionalen Element eine Menge dekomponierter intentionaler Elemente.

$$Decomp = \{\langle e_O, E_U \rangle \mid e_O \in E, E_U \subset E\}$$

Jedes Oberelement muss mindestens zwei Unterelemente besitzen. Das heißt:

$$\forall \langle e_O, e_O.sub \rangle \in Decomp : |e_O.sub| \geq 2$$

Contributions relationieren zwei intentionale Elemente zueinander. Anders als Dekompositionen besitzen sie entweder einen Wert zwischen -100 und 100 oder ein Label. Bei den Labeln werden Make, SomePositive, Help, Unknown, Hurt, SomeNegative, Break unterschieden.

Das heißt:

$$Con = \{\langle e_1, e_2, v\rangle \mid e_1, e_1 \in E, v \in V\}$$

$$V = V^{Quan} \cup V^{Qual}$$

$$V^{Quan} = \{x \in \mathbb{Z} | -100 \leq x \leq 100\}$$

$$V^{Qual} = \{Make, SomePositive, Help, Unknown, Hurt, SomeNegative, Break\}$$

Dependencies drücken Abhängigkeiten zwischen zwei Akteuren oder deren intentionalen Elementen aus. Das heißt:

$$Depend = \{\langle d_1, d_2\rangle \mid d_1, d_2 \in A \cup E \wedge (d_1 \neq d_2) \wedge (\alpha(d_1) \neq \alpha(d_2))\}$$

Abbildung 2.1 stellt die wichtigsten Konzepte von GRL-Zielmodellen grafisch dar. Gezeigt wird ein Akteur mit seinen intentionalen Elementen. Ein Akteur wird als Kreis wiedergegeben. Mithilfe der Akteursgrenze, die durch eine gestrichelte Linie dargestellt ist, können die zu dem Akteur gehörigen intentionalen Elemente definiert werden. Die verschiedenen intentionalen Elemente werden durch unterschiedliche Formen differenziert. Ziele werden durch Rechtecke mit abgerundeten Ecken dargestellt, Softgoals durch Rechtecke mit geschwungenen Kanten, Tasks durch Sechsecke, Ressourcen durch Rechtecke und Beliefs durch Ellipsen. Dekompositionen werden durch Linien angezeigt, die einen kurzen Querstrich und die Art der Dekomposition am Anfang, d. h. in der Nähe des Oberelements, aufweisen. Contributions sind durch Pfeile ausgehend von dem beeinflussenden Element zu dem beeinflussten Element gekennzeichnet. Sie werden mit dem Wert der Contribution oder einem diesbezüglichen Symbol veranschaulicht.

2.2.2 Erfüllbarkeitsprüfung

Für die Erfüllbarkeitsprüfung bei GRL-Zielmodellen stehen quantitative, qualitative und hybride Ansätze zur Auswahl (Amyot et al. 2010). Bei der Erfüllbarkeitsprüfung werden den intentionalen Elementen Werte zugewiesen. Basierend auf diesen zugewiesenen Werten können Konflikte erkannt und die Erfüllung der intentionalen Elemente automatisiert berechnet werden. Die Voraussetzung hierfür ist, dass das Zielmodell keinen Zyklus enthält.

2.2.2.1 Quantitative Erfüllbarkeitsprüfung

Bei der quantitativen Erfüllbarkeitsprüfung werden den Blattelementen (d. h. den intentionalen Elementen, die nicht weiter verfeinert werden und auf die keine Dependency- oder Contribution-Beziehung gerichtet ist) Werte zwischen -100 und 100 zugewiesen, die den Erfüllbarkeitsgrad repräsentieren. Der Wert eines Oberelements errechnet sich aus den Werten

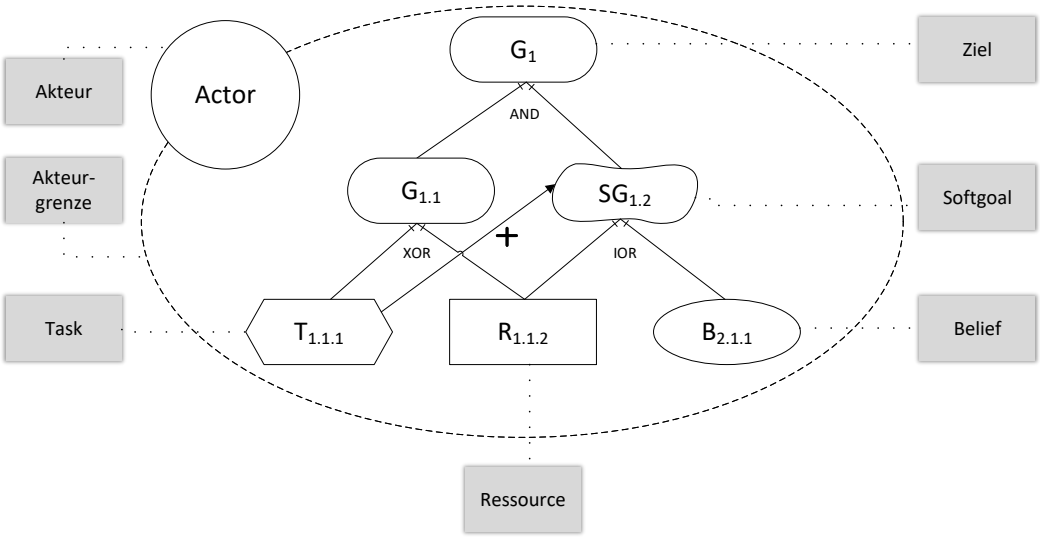


Abbildung 2.1: GRL-Zielmodell – Beispiel

der Unterelemente und der jeweiligen Art der Verbindung, die die Ober- und Unterelemente miteinander verknüpft. So ergibt sich beispielsweise der Wert eines Oberelements einer UND-Dekomposition aus dem kleinsten Wert seiner Unterelemente. Bei Contributions wird der Wert des unterstützenden Elements mit dem Wert der Contributionverbindung multipliziert und für alle eingehenden Contributionverbindungen addiert. Daher ist es bei der quantitativen Erfüllbarkeitsprüfung notwendig, dass alle Contributions mit einem quantitativen Wert versehen sind. Vorher definierte Schwellwerte verhindern, dass ein Wert von 100 oder -100 erreicht wird, wenn keine eingehende Contribution diesen allein erreicht. Bei Dependencies ergibt sich der Erfüllbarkeitsgrad des abhängigen Elements aus dem kleinsten Wert der Elemente, von dem das Element abhängig ist.

2.2.2.2 Qualitative Erfüllbarkeitsprüfung

Bei der qualitativen Erfüllbarkeitsprüfung wird intentionalen Elementen statt konkreter Werte eins von sieben verschiedenen Labels zugewiesen, die eine festgelegte Reihenfolge besitzen. Ausgehend vom höchsten Erfüllbarkeitsgrad sind dies *Satisfied*, *WeaklySatisfied*, *None*, *WeaklyDenied*, *Undecided*, *Conflict* und *Denied*, wobei Undecided und Conflict als gleichwertig betrachtet werden. Für die Berechnung des Erfüllbarkeitsgrades des Oberelements ausgehend von dem Erfüllbarkeitsgrad des Unterelements und der Art der Verbindung, die Ober- und Unterelemente verknüpfen, existieren verschiedene Regeln. So ergibt sich beispielsweise der Erfüllbarkeitsgrad eines Oberelements einer UND-Dekomposition aus dem niedrigsten Erfüllbarkeitsgrad seiner Unterelemente. Bei Contributions können qualitative Label zur qualitativen Erfüllbarkeitsprüfung eingesetzt werden. Der Erfüllbarkeitsgrad ergibt sich aus den Erfüllbarkeitsgraden der unterstützenden Elemente und der Art der Contribution. Wenn beispielsweise

das unterstützende Element *Satisfied* ist und es sich bei der Contribution um den Typ *Help* handelt, wird dem unterstützten Element der Erfüllbarkeitsgrad *WeaklySatisfied* zugewiesen. Bei mehreren eingehenden Contributions werden die verschiedenen Erfüllbarkeitsgrade nach vorgeschriebenen Regeln kombiniert. So geht beispielsweise aus einem *WeaklySatisfied* und einem *Denied* ein *WeaklyDenied* hervor. Bei Dependencies erhält das abhängige Element den niedrigsten Erfüllbarkeitsgrad der Elemente, von denen es abhängig ist.

2.2.2.3 Hybride Erfüllbarkeitsprüfung

Die hybride Erfüllbarkeitsprüfung kombiniert die quantitative und die qualitative Erfüllbarkeitsprüfung. Sie erlaubt es, Elementen quantitative Erfüllbarkeitsgrade zuzuweisen, auch wenn die Contributions qualitative Label besitzen. Dazu werden die Label folgendermaßen in Werte umgerechnet: Make (100), SomePositive (75), Help (25), Unknown (0), Hurt (-25), SomeNegative (-75), Break (-100).

2.3 Kardinalitätsbasierte Feature-Modelle

Feature-Modelle, die ursprünglich im Rahmen der Methode der Feature-Oriented Domain Analysis (FODA) (Kang et al. 1990) vorgeschlagen wurden, sind die bekannteste und am weitesten verbreitete Modellierungssprache zur Dokumentation von Variabilität in Produktlinien (vgl. (Berger et al. 2013)). Ein Feature-Modell stellt eine Menge von Features, d. h. ein für den Endbenutzer sichtbares Merkmal eines Systems (Pohl et al. 2005), als hierarchische Baumstruktur dar. Diese Baumstruktur entsteht, da Features durch Beziehungen zwischen einem Eltern-Feature und seinen Kind-Features zerlegt werden können. Ein Feature kann obligatorisch oder optional sein; obligatorische Features charakterisieren die Gemeinsamkeiten zwischen allen Produkten. Darüber hinaus können zwischen den Features eines Feature-Modells Constraints definiert werden, die typischerweise Requires- und Excludes-Beziehungen beschreiben.

2.3.1 Definition von kardinalitätsbasierten Feature-Modellen

Ein Feature-Modell FM ist in Schobbens et al. (2006) definiert als:

$$FM = \langle N, r, P, \lambda, DE, CE, \phi \rangle$$

wobei N die Menge der Knoten, r die Wurzel, P die Menge der primitiven Knoten, λ eine Relationsfunktion, DE die Menge der Dekompositionskanten, CE die Menge der Constraintkanten und ϕ textuelle Constraints sind. Die Wurzel wird auch als Konzept bezeichnet. Es gilt:

$$r \in N$$

Nur die Wurzel r besitzt keinen Elternknoten. Es gilt:

$$\forall n \in N : (\nexists n' \in N : n' \rightarrow n) \Leftrightarrow n = r$$

Primitive Knoten stellen Features dar, die sich auf das Endprodukt auswirken. Es gilt:

$$P \subseteq N$$

λ weist Knoten einen Operator zu. Es gilt:

$$\lambda \subseteq N \times NT$$

wobei NT eine Menge boolescher Funktionen ist. Es gilt:

$$NT = and \cup or \cup opt \cup vp(i..j)$$

wobei and die Menge der Funktionen and_s ist, von denen es für jede Arität [4] s genau eine gibt, die *wahr* ist, wenn alle s Argumente *wahr* sind. or ist die Menge der Funktionen or_s, von denen es für jede Arität s genau eine Funktion gibt, die *wahr* ist, wenn mindestens ein s Argument *wahr* ist. opt ist die Menge der Funktionen opt_s, von denen es für jede Arität s genau eine gibt, die immer *wahr* ist. $vp(i..j)$ ist die Menge der Funktionen $vp_s(i..j)$, von denen es für jede Arität s genau eine gibt, die *wahr* ist, wenn mindestens i und maximal j der s Argumente *wahr* sind.

Dekompositionskanten DE verbinden Eltern- und Kind-Features. Es gilt:

$$DE \subseteq N \times N$$

$(n, n') \in DE$ wird als $n \to n'$ geschrieben. Die Dekompositionskanten müssen azyklisch sein. Daher gilt:

$$\nexists n_1, ..., n_k \in N : n_1 \to ... \to n_k \to n_1$$

Constraintkanten CE verbinden zwei Features, zwischen denen Abhängigkeiten bestehen. Es gilt:

$$CE \subseteq N \times GCT \times N$$

wobei GCT ein binärer boolescher Operator ist. Es gilt:

$$GCT = \{Requires, Mutex\}$$

Weitere Abhängigkeiten zwischen Features lassen sich durch logische Ausdrücke beschreiben. Hierzu wird ϕ als eine Menge boolescher Formeln definiert, bei denen die Prädikate die Knoten des Feature-Modells sind.

Abbildung 2.2 zeigt die grafische Darstellung der wichtigsten Konzepte von Feature-Modellen. Zu sehen ist ein Feature-Modell mit sieben verschiedenen Features, dargestellt durch Rechtecke. Der nicht ausgefüllte Kreis an der Dekompositionskante zeigt, dass das Feature F_1 optional ist. Die Features F_2 und F_3 sind immer erforderlich, dargestellt durch den ausgefüllten Kreis an

[4] Mit Arität wird die Anzahl der jeweiligen Kinderknoten bezeichnet. So wird bspw. einem Knoten mit drei Kinderknoten die Funktion and_3 zugewiesen.

der Dekompositionskante. Die Features $F_{1.1}$ und $F_{1.2}$ sind Kind-Features von F_1. Der Variationspunkt von F_1 definiert, dass entweder eines oder beide ausgewählt werden müssen, wenn Feature F_1 ausgewählt ist. Die Features $F_{3.1}$ und $F_{3.2}$ sind Kind-Features von F_3. Es muss entweder Feature $F_{3.1}$ oder Feature $F_{3.2}$ ausgewählt werden. Die Constraintkante spezifiziert, dass das Feature $F_{3.1}$ ebenfalls ausgewählt werden muss, wenn das Feature $F_{1.2}$ ausgewählt wurde.

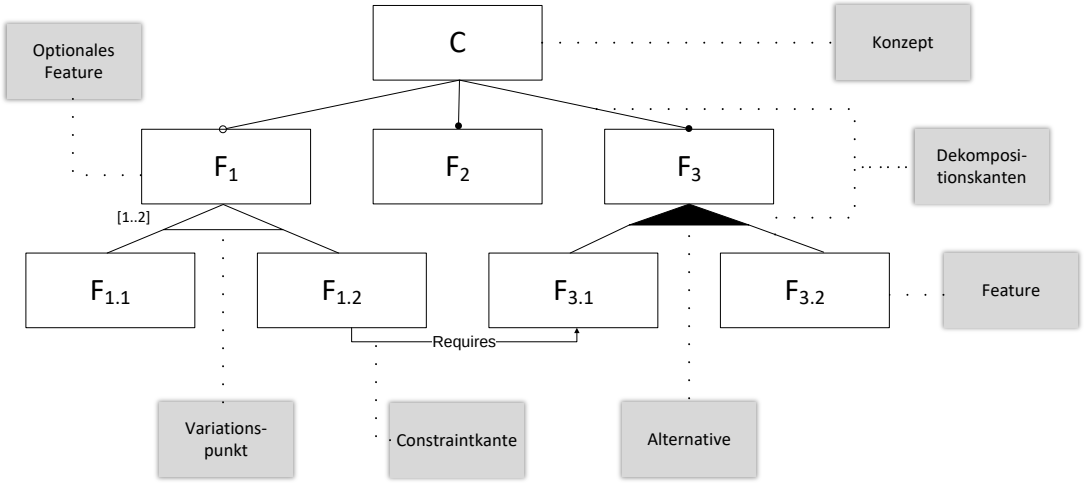


Abbildung 2.2: Feature-Modell – Beispiel

Kardinalitäts-basierte Feature-Modelle (Czarnecki et al. 2005) erweitern reguläre Feature-Modelle (Kang et al. 1990), um die Anzahl gleichartiger Features in Intervallen spezifizieren zu können. Czarnecki et al. (2004 2005) schlagen zwei Erweiterungen gegenüber konventionellen Feature-Modellen vor: die Feature-Kardinalität und die Feature-Group-Kardinalität. Feature-Kardinalitäten definieren ein Intervall i_F, wie oft ein Feature $n \in N$ (und seine Kind-Features, d. h. ein Teilbaum) instanziiert werden können. Die Relation κ_F weist jedem Feature n eine Kardinalität $i_F \in I_F$ zu.

$$\kappa_F : N \rightarrow I_F$$

Eine Feature-Kardinalität ist ein Intervall $i_F \in I_F$, bei dem m_F^{min} die Mindestanzahl der Features und m_F^{max} die Maximalanzahl der Features bezeichnet. Dabei kann die Maximalanzahl unbeschränkt sein, was durch das Symbol * ausgedrückt wird. Es gilt:

$$0 \leq m_F^{min} \leq m_F^{max}$$

Feature-Group-Kardinalitäten geben an, wie viele unterschiedliche Features einer Gruppe von Features ausgewählt werden müssen. Die Relation κ_T weist jeder Feature Group vp eine Kardinalität $i_T \in I_T$ zu.

$$\kappa^T : VP \rightarrow I^T$$

Eine Feature-Group-Kardinalität ist ein Intervall I_T, bei dem m_T^{min} die Mindestanzahl der gruppierten Features und m_T^{max} die Maximalanzahl der gruppierten Features bezeichnet. Bei k unterschiedlichen Features gilt:

$$k > 0$$

$$0 \leq m_T^{min} \leq m_T^{max} \leq k$$

In Dhungana et al. (2011) werden diese Feature-Group-Kardinalitäten als Feature-Group-Typ-Kardinalitäten bezeichnet und von Feature-Group-Instanz-Kardinalitäten unterschieden.

Die Feature-Group-Instanz-Kardinalität gibt an, wie viele Features aus einer Feature Group mindestens und höchstens instanziiert werden müssen. Die Relation κ_I weist jeder Feature Group vp_k eine Kardinalität $i_I \in I_I$ zu.

$$\kappa_I : VP \rightarrow I_I$$

Eine Feature-Group-Instanz-Kardinalität ist ein Intervall $i_I \in I_I$, bei dem m_I^{min} die Mindestanzahl der Features und m_I^{max} die Maximalanzahl der Features aus der gleichen Feature Group bezeichnet. Die Untergrenze des Intervalls muss mindestens so hoch sein wie die Summe der Untergrenzen der Kardinalitäten der Kinder-Feature. Die Obergrenze des Intervalls darf nicht höher sein als die Summe der Obergrenzen der Kardinalitäten der Kinder-Feature. Es gilt:

$$(\sum_{\forall n' : n \rightarrow n'} m_F^{min}) \leq m_I^{min} \leq m_I^{max} \leq (\sum_{\forall n' : n \rightarrow n'} m_F^{max})$$

Eine andere Erweiterung von Dhungana et al. (2011) bezieht sich auf Constraints. Requires-Constraints können auch mit Kardinalitäten $i_R \in I_R$ annotiert werden, um auszudrücken, dass die Auswahl eines bestimmten Features eine bestimmte Anzahl von Instanzen eines anderen Features erfordert. Die Relation κ_R weist jeder Relation ce eine Kardinalität $i_R \in I_R$ zu.

$$\kappa_R : CE \rightarrow I_R$$

Abbildung 2.3 zeigt die grafische Darstellung der verschiedenen Arten von Kardinalitäten in Feature-Modellen. Gezeigt werden Feature-Kardinalitäten, Feature-Group-Typ-Kardinalitäten und Feature-Group-Instanz-Kardinalitäten. Die Feature-Kardinalität [0..*] bei Feature F_1 gibt an, dass dieses Feature gar nicht bis beliebig oft ausgewählt werden kann. Die Feature-Kardinalität [1..*] bei Feature F_2 und F_3 gibt an, dass diese Features mindestens einmal ausgewählt werden müssen und beliebig oft ausgewählt werden können. Daher ist es nicht mehr erforderlich, Optionalität an den Dekompositionskanten darzustellen. Die Feature-Group-Typ-Kardinalität gibt an, dass entweder Feature $F_{1.1}$ oder $F_{1.2}$ oder beide ausgewählt werden müssen. Allerdings wird nicht eingeschränkt, wie oft diese Features insgesamt ausgewählt werden können. Die Feature-Group-Instanz-Kardinalität spezifiziert, dass von den Features $F_{3.1}$ und $F_{3.2}$ insgesamt zwei bis fünf ausgewählt werden können. Dabei ist es unerheblich, wie viele von den $F_{3.1}$ und $F_{3.2}$ selbst ausgewählt werden, so lange die Anzahl beider Features zusammen zwischen zwei

und fünf liegt. Die Constraintkante spezifiziert, dass das Feature $F_{3.1}$ ebenfalls mindestens einmal ausgewählt werden muss, wenn das Feature $F_{1.2}$ ausgewählt wurde.

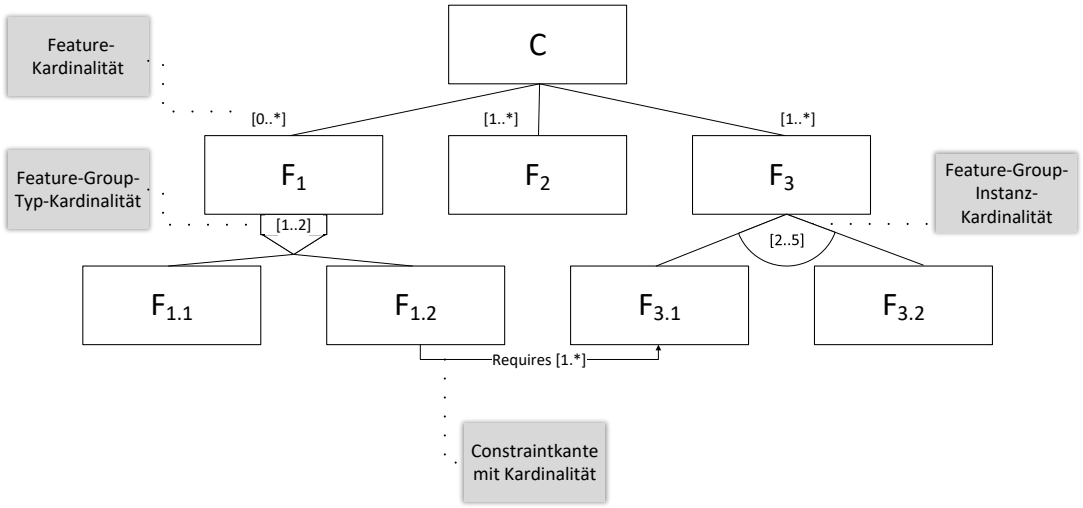


Abbildung 2.3: Kardinalitätsbasiertes Feature-Modell – Beispiel

2.3.2 Anomalien in kardinalitätsbasierten Feature-Modellen

Kardinalitätsbasierte Feature-Modelle können Anomalien aufweisen, die gegebenenfalls dazu führen, dass das Modell aufgrund von Widersprüchen zwischen Kardinalitäten unbrauchbar ist (Weckesser et al. 2016). Es existieren zwei Arten von Anomalien, die in kardinalitätsbasierten Feature-Modellen auftreten können: ungültige Unter- und Obergrenzen sowie Intervalllücken. Ungültige Unter- und Obergrenzen entstehen, wenn eine Kardinalität definiert wird, die mehr Instanzen zulässt, als aufgrund von Einschränkungen möglich sind. Wenn zum Beispiel eine Feature-Group-Instanz-Kardinalität mindestens eine Instanz erfordert, die beiden Kind-Features jedoch selbst beide mindestens eine Instanz erfordern, ist die Untergrenze der Feature-Group-Instanz-Kardinalität ungültig, da sie mindestens zwei betragen müsste. Ähnlich verhält es sich bei ungültigen Obergrenzen. Eine solche Anomalie liegt beispielsweise vor, wenn die Summe der Maximal-Kardinalitäten aller Kind-Features kleiner ist als die maximale Feature-Group-Instanz-Kardinalität des Eltern-Features. Dies trifft auch zu, wenn die Kardinalität unbeschränkt ist. Ungültige Unter- und Obergrenzen in kardinalitäsbasierten Feature-Modellen lassen sich per Integer Linear Programming (ILP) automatisiert ermitteln. Dazu wird das kardinalitätsbasierte Feature-Modell in eine Menge von Ungleichungen kodiert, die von einem ILP-Solver gelöst werden. Dadurch lassen sich die gültige Unter- oder Obergrenze angeben.

Intervalllücken bezeichnen tote, d. h. nicht instanziierbare, Kardinalitäten innerhalb eines Kardinalitätsintervalls. Ähnlich wie ungültige Unter- und Obergrenzen entstehen Intervalllücken durch widersprüchliche Kardinalitäten. Beispielsweise tritt eine Intervalllücke auf,

wenn ein Exclude-Constraint bestimmte Kardinalitäten ausschließt. Intervallücken können durch Satisfiability Modulo Theories (SMT)-Solver automatisiert erkannt werden.

Ein Tool, das sowohl die Analyse von Unter- und Obergrenzen als auch die Analyse von Intervalllücken unterstützt, wird in Schnabel et al. (2016) vorgestellt.

2.4 Validierungsansätze für Anforderungen

Zur Validierung von Anforderungen existieren Ansätze, die die manuelle Suche nach Defekten in Spezifikationen unterstützen. Diese Ansätze werden als Review- oder Inspektionsansätze bezeichnet, wobei sich Inspektionsansätze häufig durch einen höheren Grad der Formalität als Reviewansätze auszeichnen (Spillner und Linz 2019). Software-Inspektionsansätze sind durch ihren Inspektionsprozess charakterisiert, der verschiedene Phasen der Inspektion für Planung, Vorbereitung, Überprüfung, Inspektionssitzungen sowie Überarbeitung und Nachbereitung definiert. Wesentlich ist auch die Definition der spezifischen Rollen, die die verschiedenen Reviewer oder Inspektoren während einer Inspektion einnehmen.

Neben der Definition eines Inspektionsprozesses und der Rollen haben Aurum et al. (2002) sowie eine spätere Umfrage von Kollanus und Koskinen (2009) Hilfslesetechniken identifiziert. Diese Techniken legen fest, wie die Reviewer die Begutachtung durchführen, d. h., wie sie das zu validierende Entwicklungsartefakt lesen sollen. Dies kann z. B. unter Berücksichtigung einer Checkliste erfolgen, an die sich der Reviewer halten muss (d. h. Checklisten-basiertes Lesen), durch die Definition von Szenarien, die eine Sammlung von Prozeduren zur Erkennung bestimmter Fehlerklassen darstellen (d. h. szenariobasiertes Lesen), oder durch die Definition bestimmter Perspektiven, die ein Reviewer einnehmen soll, um sicherzustellen, dass die Spezifikation aus Sicht der Tester testbar, aus Sicht der Entwickler implementierbar usw. ist (d. h. perspektivisches Lesen).

Darüber hinaus werden Inspektionen für verschiedene zu validierende Entwicklungsartefakte definiert (bspw. Anforderungen, Design, Code, Testfälle (Aurum et al. 2002)). Daher existieren verschiedene Inspektionsansätze, die z. B. auf Code (z. B. (Almeida et al. 2003; Laitenberger 1998)) oder Anforderungsartefakte (z. B. (Miller et al. 1998; Berling und Runeson 2003; Maldonado et al. 2006)) zugeschnitten sind.

In den letzten Jahren wurden Ansätze vorgeschlagen, die diese gut etablierten Inspektions- und Lesetechniken nutzen – und anpassen –, um auch modellbasierte Entwicklungsartefakte zu berücksichtigen. Zum Beispiel berichten Conradi et al. (2003) und Laitenberger et al. (2000) über Experimente zur Inspektion von UML-Modellen. Lucia et al. (2008) stellen die Ergebnisse zweier kontrollierter Experimente mit Bachelor- und Master-Studenten vor, die zeigen, dass UML-Klassendiagramme deutlich leichter zu verstehen sind als ER-Diagramme. Bavota et al. (2011) führten eine Studie durch, um UML-Klassendiagramme und ER-Diagramme hinsichtlich ihrer Auswirkungen auf das Modellverständnis zu vergleichen. Sie kamen ebenfalls zu dem

Schluss, dass UML-Klassendiagramme im Allgemeinen leichter zu verstehen sind als ER-Diagramme.

2.5 Sichtenbildung

In der Softwareentwicklung werden Sichten genutzt, um komplexe Sachverhalte durch Abstraktion verständlicher darzustellen (Kotonya und Sommerville 1996; Finkelstein et al. 1991). Sichten sind vereinfachte Repräsentationen, die auf bestimmte Aspekte fokussieren. Um welche Aspekte es sich handelt, wird durch einen Viewpoint bestimmt. Die ISO/IEC/IEEE 42010 (2011) definiert Sicht (engl. View) und Viewpoint wie folgt:

> „4.2.4 Architecture views and viewpoints
>
> An architecture description includes one or more architecture views. An architecture view (or simply, view) addresses one or more of the concerns held by the system's stakeholders.
>
> An architecture view expresses the architecture of the system-of-interest in accordance with an architecture viewpoint (or simply, viewpoint). There are two aspects to a viewpoint: the concerns it frames for stakeholders and the conventions it establishes on views.
>
> An architecture viewpoint frames one or more concerns. A concern can be framed by more than one viewpoint.
>
> A view is governed by its viewpoint: the viewpoint establishes the conventions for constructing, interpreting and analyzing the view to address concerns framed by that viewpoint. Viewpoint conventions can include languages, notations, model kinds, design rules, and/or modelling methods, analysis techniques and other operations on views."
>
> (ISO/IEC/IEEE 42010 2011)

Sichten können durch Modelltransformationen erzeugt werden (Czarnecki und Helsen 2003; Sendall und Kozaczynski 2003). Abbildung 2.4 illustriert vier Möglichkeiten, um Sichten zu bilden.

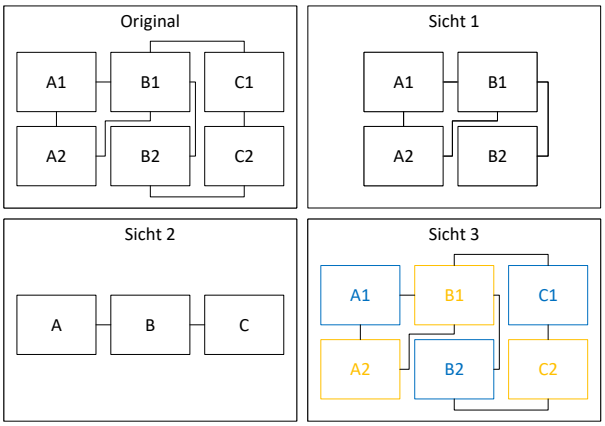


Abbildung 2.4: Abstraktion

Das Originalmodell zeigt sechs Boxen. Sicht 1 blendet die Boxen C1 und C2 aus und erlaubt dadurch die Fokussierung auf die verbleibenden Boxen. Sicht 2 zeigt drei Boxen, die jeweils zwei Boxen aus dem Originalmodell repräsentieren. Diese Sicht abstrahiert von Details und erlaubt einen einfacheren Überblick. In Sicht 3 sind alle Boxen aus dem Originalmodell eingefärbt, die unterschiedlichen Farben zeigen die Zugehörigkeit zu bestimmten Eigenschaften. In dieser Sicht ist zu erkennen, dass die Boxen A1, B2 und C1 über eine Gemeinsamkeit verfügen und die Boxen A2, B1 und C2 über eine andere.

3 Kapitel
Stand der Wissenschaft

Die Analyse des Stands der Wissenschaft zeigt, dass es in der Softwareproduktlinienentwicklung Ansätze gibt, die sich auf die Spezifikation von Zielen für Systems-of-Systems übertragen lassen. Des Weiteren existieren Ansätze zur Analyse solcher Spezifikationen, die automatisiert Fehler identifizieren, die auf Widersprüche zurückzuführen sind. Jedoch liegt bislang keine Unterstützung für die Prüfung solcher Spezifikationen gegen Stakeholder-Wünsche vor, bei der die komplexen Beziehungen zwischen Zielen und SoS-Konfigurationen so aufbereitet werden, dass die erforderliche manuelle Prüfung erleichtert wird.

Dieses Kapitel stellt den Stand der Wissenschaft vor. In Abschnitt 3.1 werden zunächst relevante Themenbereiche identifiziert, deren Ansätze einen Beitrag zu der in Abschnitt 1.5 vorgestellten Zielsetzung leisten können. Anhand des in Abschnitt 3.2 vorgestellten Bewertungsrahmens wird geprüft, ob die Ansätze aus den relevanten Themenbereichen bestimmte Kriterien erfüllen. Die Ansätze aus den als relevant identifizierten Themenbereichen werden in den Abschnitten 3.3 und 3.4 detailliert vorgestellt und bewertet. Abschnitt 3.5 zeigt daraufhin zusammenfassend auf, welche Bereiche der Zielsetzung bereits durch existierende Ansätze abgedeckt sind und welche Lücke noch besteht, die in dieser Arbeit adressiert wird.

3.1 Relevante Themenbereiche

Basierend auf der in Abschnitt 1.5 vorgestellten Zielsetzung ergeben sich diverse Bereiche, die einen Beitrag zur Bearbeitung der Problemstellung leisten können. Es handelt sich unter anderem um Ansätze, die sich mit der Spezifikation beschäftigen. Von Interesse sind hier vor allem Ansätze

- zur Spezifikation von Zielen sowie
- zur Spezifikation von Systemen, die unterschiedliche Systemzusammensetzungen aufweisen können.

Bei Systemen, die unterschiedliche Systemzusammensetzungen aufweisen können, ist ferner zu beachten, dass dies nicht nur auf Systems-of-Systems zutrifft, sondern auch auf andere Arten

von Systemen wie bspw. Produktlinien, Komponenten-basierte Systeme und Webservices, die somit ebenfalls von Interesse sind. Besonders aufschlussreich sind außerdem Spezifikationsansätze, die beide Punkte berücksichtigen, also zielorientierte Spezifikationsansätze für variable Systeme.

Des Weiteren sind Ansätze informativ, die sich mit der Valdidierung beschäftigen, darunter Ansätze

- zur Validierung von Zielspezifikationen sowie
- zur Validierung von Entwicklungsartefakten für Systeme, die unterschiedliche Systemzusammensetzungen aufweisen können.

Auch hier werden nicht nur Validierungsansätze für Systems-of-Systems berücksichtigt, sondern auch alle andere Systeme, die unterschiedliche Systemzusammensetzungen aufweisen können.

Außerdem von Interesse sind Ansätze, die automatisiert Inkonsistenzen in

- Zielmodellen
- SoS-Modellen
- zwischen Ziel- und SoS-Modellen

identifizieren können, da sich hierdurch bereits manche Defekte automatisiert identifizieren lassen.

3.2 Bewertungsrahmen

Zur Bewertung der Ansätze aus den relevanten Themenbereichen wurden basierend auf der in Abschnitt 1.5 vorgestellten Zielsetzung die Bewertungskriterien definiert.

Dabei wird zwischen den Themenbereichen Spezifikation und Validierung unterschieden, für die jeweils ein Bewertungsrahmen vorliegt.

3.2.1 Spezifikation von Zielen für SoS-Konfigurationen

Bei Spezifikationsansätzen wird betrachtet, ob Ziele berücksichtigt werden und unterschiedliche SoS-Konfigurationen und die Relation zwischen Zielen und SoS-Konfigurationen Beachtung finden. Ein weiterer wichtiger Punkt beschäftigt sich damit, ob die Ansätze Unterstützung bieten, um den Umfang der Spezifikation zu reduzieren und so auch die Entwicklung von Systems-of-Sytems mit einer großen Menge an SoS-Konfigurationen zu unterstützen. Dies ist notwendig, damit Entwickler die Spezifikation lesen, verstehen und überblicken können. Daher werden geeignete Mechanismen zur Komplexitätsreduktion benötigt. Daraus ergeben sich für Spezifikationsansätze die folgenden vier Bewertungskriterien:

1. *Kriterium S1:* Berücksichtigung von Zielen bei der Spezifikation
2. *Kriterium S2:* Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Spezifikation
3. *Kriterium S3:* Berücksichtigung der Relation von Zielen und SoS-Konfigurationen bei der Spezifikation
4. *Kriterium S4:* Reduktion des Umfangs der zu erstellenden Spezifikation

Eine wichtige Eigenschaft eines Ansatzes, um zur Zielsetzung der Arbeit beizutragen, ist die Fähigkeit, die Spezifikation von Zielen zu unterstützen. Dafür wird das Bewertungskriterium S1 definiert, das sich mit der Spezifizierbarkeit von Zielen beschäftigt. Unterschieden werden die drei in Tabelle 3.1 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Ziele zu spezifizieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht die Spezifikation von Zielen erlaubt, aber dennoch einen Beitrag zur Spezifikation von Zielen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz die Spezifikation von Zielen nicht erlaubt und auch keinen Beitrag zur Spezifikation von Zielen leisten kann.

Tabelle 3.1: Bewertungskriterium S1

erfüllt	Der Ansatz erlaubt die Spezifikation von Zielen.
teilweise erfüllt	Der Ansatz erlaubt die Spezifikation von Zielen nicht, kann aber trotzdem einen Beitrag leisten.
nicht erfüllt	Der Ansatz erlaubt die Spezifikation von Zielen nicht und und kann auch keinen Beitrag leisten.

Eine weitere wichtige Eigenschaft eines Ansatzes, der einen Beitrag zur Zielerreichung dieser Arbeit leisten kann, ist die Fähigkeit, die Spezifikation von unterschiedlichen SoS-Konfigurationen zu unterstützen. Dafür wird das Bewertungskriterium S2 definiert, das sich mit der Spezifizierbarkeit von unterschiedlichen SoS-Konfigurationen beschäftigt. Differenziert wird zwischen den drei in Tabelle 3.2 vorgestellten Erfüllungsgraden. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Ziele zu spezifizieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht die Spezifikation von unterschiedlichen SoS-Konfigurationen erlaubt, aber dennoch einen Beitrag zur Spezifikation von unterschiedlichen SoS-Konfigurationen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz die Spezifikation von unterschiedlichen SoS-Konfigurationen nicht erlaubt und auch keinen Beitrag zur Spezifikation von unterschiedlichen SoS-Konfigurationen leisten kann.

Als weitere wichtige Eigenschaft eines Ansatzes, um die Zielsetzung dieser Arbeit zu adressieren, ist die Fähigkeit, die Spezifikation der Relation von Zielen und unterschiedlichen SoS-Konfigurationen zu unterstützen, um zu definieren, welche Ziele für welche SoS-Konfiguration

Tabelle 3.2: Bewertungskriterium S2

erfüllt	Der Ansatz erlaubt die Spezifikation von unterschiedlichen Systemzusammensetzungen.
teilweise erfüllt	Der Ansatz erlaubt die Spezifikation von unterschiedlichen Systemzusammensetzungen nicht, kann aber trotzdem einen Beitrag leisten.
nicht erfüllt	Der Ansatz erlaubt die Spezifikation von unterschiedlichen Systemzusammensetzungen nicht und kann auch keinen Beitrag leisten.

gewünscht sind. Dafür wird das Bewertungskriterium S3 definiert, das sich mit der Spezifizierbarkeit von der Relation von Zielen und unterschiedlichen SoS-Konfigurationen beschäftigt. Es erfolgt eine Abgrenzung der drei in Tabelle 3.3 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, die Relation von Zielen und unterschiedlichen SoS-Konfigurationen zu spezifizieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht die Spezifikation der Relation von Zielen und unterschiedlichen SoS-Konfigurationen erlaubt, aber dennoch einen Beitrag zur Spezifikation der Relation von Zielen und unterschiedlichen SoS-Konfigurationen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz die Spezifikation der Relation von Zielen und unterschiedlichen SoS-Konfigurationen nicht erlaubt und auch keinen Beitrag zur Spezifikation der Relation von Zielen und unterschiedlichen SoS-Konfigurationen leisten kann.

Tabelle 3.3: Bewertungskriterium S3

erfüllt	Der Ansatz erlaubt die Spezifizierung der Relation von Zielen und unterschiedlichen Systemzusammensetzungen.
teilweise erfüllt	Der Ansatz erlaubt keine Spezifizierung der Relation von Zielen und unterschiedlichen Systemzusammensetzungen, kann aber dennoch einen Beitrag leisten.
nicht erfüllt	Der Ansatz erlaubt die Relation von Zielen und unterschiedlichen Systemzusammensetzungen nicht und kann auch keinen Beitrag leisten.

Ebenfalls ist als wichtige Eigenschaft eines Ansatzes, der zur Zielsetzung der Arbeit beitragen kann, die Fähigkeit zu nennen, den Umfang der Spezifikation zu reduzieren, um auch die Entwicklung von Systems-of-Systems mit mehr unterschiedlichen SoS-Konfigurationen zu unterstützen, als manuell handhabbar ist. Dafür wird das Bewertungskriterium S4 definiert, das sich mit der Reduktion des Umfangs der Spezifikation beschäftigt. Hier werden die drei in Tabelle 3.4 vorgestellten Erfüllungsgrade unterschieden. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, den Umfang der Spezifikation zu reduzieren. Das Kriterium ist teilweise

erfüllt, wenn der Ansatz zwar nicht erlaubt, den Umfang der Spezifikation zu reduzieren, aber dennoch einen Beitrag zur Reduktion des Umfangs der Spezifikation leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz es nicht erlaubt, den Umfang der Spezifikation zu reduzieren und auch keinen Beitrag zur Reduktion des Umfangs der Spezifikation leisten kann.

Tabelle 3.4: Bewertungskriterium S4

erfüllt	Der Ansatz erlaubt die Relationierung der Ziele und Systemzusammensetzungen, ohne dass für jede Systemzusammensetzung die Ziele separat spezifiziert werden müssen.
teilweise erfüllt	Der Ansatz erlaubt keine Relationierung der Ziele und Systemzusammensetzungen, ohne dass für jede Systemzusammensetzung die Ziele separat spezifiziert werden müssen, kann aber dennoch einen Beitrag leisten.
nicht erfüllt	Der Ansatz erlaubt keine Relationierung der Ziele und Systemzusammensetzungen, ohne dass für jede Systemzusammensetzung die Ziele separat spezifiziert werden müssen, und kann auch keinen Beitrag leisten.

3.2.2 Validierung von Zielspezifikationen für SoS-Konfigurationen

Zur Beurteilung der Unterstützung der Validierung von Zielspezifikationen für SoS-Konfigurationen werden die folgenden acht Bewertungskriterien definiert:

1. *Kriterium V1:* Berücksichtigung von Zielen bei der Validierung
2. *Kriterium V2:* Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung
3. *Kriterium V3:* Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist
4. *Kriterium V4:* Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind
5. *Kriterium V5:* Darstellung
6. *Kriterium V6:* Unterstützung der Validierung durch die Erkennung von Inkonsistenzen im Zielmodell
7. *Kriterium V7:* Unterstützung der Validierung durch die Erkennung von Inkonsistenzen im SoS-Modell

8. *Kriterium V8:* Unterstützung der Validierung durch die Erkennung von Inkonsistenzen zwischen dem Ziel- und SoS-Modell

Eine wichtige Eigenschaft eines Ansatzes, um zur Zielsetzung der Arbeit beizutragen, ist die Fähigkeit, die Validierung von Zielen zu unterstützen. Dafür wird das Bewertungskriterium V1 definiert, das sich mit der Validierung von Zielen beschäftigt. Es wird eine Unterscheidung zwischen den drei in Tabelle 3.5 vorgestellten Erfüllungsgraden getroffen. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Ziele zu validieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht die Validierung von Zielen erlaubt, aber dennoch einen Beitrag zur Validierung von Zielen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz die Validierung von Zielen nicht erlaubt und auch keinen Beitrag zur Validierung von Zielen leisten kann.

Tabelle 3.5: Bewertungskriterium V1

erfüllt	Der Ansatz erlaubt die Validierung von Zielen.
teilweise erfüllt	Der Ansatz erlaubt die Validierung von Zielen nicht, kann aber dennoch einen Beitrag leisten..
nicht erfüllt	Der Ansatz erlaubt die Validierung von Zielen nicht und kann auch keinen Beitrag leisten.

Als weitere wichtige Eigenschaft eines Ansatzes, der zur Zielerreichung der Arbeit beiträgt, ist die Fähigkeit, die Validierung von unterschiedlichen SoS-Konfigurationen zu unterstützen. Dafür wird das Bewertungskriterium V2 definiert, das sich mit der Validierung von unterschiedlichen SoS-Konfigurationen beschäftigt. Differenziert werden die drei in Tabelle 3.6 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Ziele zu validieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht die Validierung von unterschiedlichen SoS-Konfigurationen erlaubt, aber dennoch einen Beitrag zur Validierung von unterschiedlichen SoS-Konfigurationen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz die Validierung von unterschiedlichen SoS-Konfigurationen nicht erlaubt und auch keinen Beitrag zur Validierung von unterschiedlichen SoS-Konfigurationen leisten kann.

Des Weiteren ist als wichtige Eigenschaft eines Ansatzes, der einen Beitrag zur Zielerreichung dieser Arbeit leistet, die Fähigkeit zu nennen, dass die Validierung der Relation von Zielen und unterschiedlichen SoS-Konfigurationen unterstützt wird, um zu validieren, welche Ziele für welche SoS-Konfiguration gewünscht sind. Dafür werden die Bewertungskriterien V3 und V4 definiert. V3 beschäftigt sich mit der Frage, ob der Ansatz Unterstützung bei der Prüfung bietet, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist. Unterschieden werden die drei in Tabelle 3.7 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz eine Unterstützung bei der Prüfung bietet, ob ein Ziel für genau

Tabelle 3.6: Bewertungskriterium V2

erfüllt	Der Ansatz erlaubt die Validierung von unterschiedlichen Systemzusammensetzungen.
teilweise erfüllt	Der Ansatz erlaubt die Validierung von unterschiedlichen Systemzusammensetzungen nicht, kann aber trotzdem einen Beitrag leisten.
nicht erfüllt	Der Ansatz erlaubt die Validierung von unterschiedlichen Systemzusammensetzungen nicht und kann auch keinen Beitrag leisten.

alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar keine direkte Unterstützung bei der Prüfung bietet, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist, aber dennoch einen Beitrag zur Prüfung leisten kann, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist. Das Kriterium ist nicht erfüllt, wenn der Ansatz keine Unterstützung bei der Prüfung bietet, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist.

Tabelle 3.7: Bewertungskriterium V3

erfüllt	Der Ansatz unterstützt die Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist.
teilweise erfüllt	Der Ansatz unterstützt nicht direkt die Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist, kann aber trotzdem einen Beitrag leisten
nicht erfüllt.	Der Ansatz unterstützt nicht die Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert ist, für die es gewünscht ist.

Bewertungskriterium V4 beschäftigt sich mit der Frage, ob der Ansatz Unterstützung bei der Prüfung bietet, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind. Es werden die drei in Tabelle 3.8 vorgestellten Erfüllungsgrade unterschieden. Das Kriterium ist erfüllt, wenn der Ansatz eine Unterstützung bei der Prüfung bietet, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar keine direkte Unterstützung bei der Prüfung bietet, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind, aber dennoch ein Beitrag zur Prüfung geleistet wird, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind. Das Kriterium ist nicht erfüllt, wenn der Ansatz keine Unterstützung bei der Prüfung bietet, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind.

Tabelle 3.8: Bewertungskriterium V4

erfüllt	Der Ansatz unterstützt die Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind.
teilweise erfüllt	Der Ansatz unterstützt nicht direkt die Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind, kann aber trotzdem einen Beitrag leisten.
nicht erfüllt	Der Ansatz unterstützt nicht die Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind, die für diese SoS-Konfiguration gewünscht sind.

Eine weitere wichtige Eigenschaft eines Ansatzes, um zur Zielsetzung der Arbeit beizutragen, ist die Fähigkeit, die Darstellung der zu validierenden Spezifikation zu vereinfachen. Durch eine vereinfachte Darstellung können Fehler in der Spezifikation leichter erkannt werden. Dafür wird das Bewertungskriterium V5 definiert, das sich mit der vereinfachten Darstellung der zu validierenden Spezifikation beschäftigt. Unterschieden werden die drei in Tabelle 3.9 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz die Darstellung der Spezifikation zur Unterstützung der Validierung vereinfacht. Das Kriterium ist teilweise erfüllt, wenn der Ansatz zwar nicht direkt die Darstellung vereinfacht, dennoch aber einen Beitrag zur Vereinfachung der Darstellung leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz nicht die Darstellung zur Unterstützung der Validierung vereinfacht und auch keinen Beitrag zur Vereinfachung der Darstellung der Spezifikation leisten kann.

Tabelle 3.9: Bewertungskriterium V5

erfüllt	Der Ansatz vereinfacht die Darstellung der Spezifikation zur Unterstützung der Validierung.
teilweise erfüllt	Der Ansatz vereinfacht nicht die Darstellung der Spezifikation zur Unterstützung der Validierung, kann aber trotzdem einen Beitrag leisten.
nicht erfüllt	Der Ansatz vereinfacht nicht die Darstellung der Spezifikation zur Unterstützung der Validierung und kann auch keinen Beitrag leisten.

Zu den Eigenschaften eines Ansatzes, um zur Zielerreichung dieser Arbeit beizutragen, gehört die Fähigkeit, Inkonsistenzen im Zielmodell zu identifizieren, um die Validierung der Zielspezifikation zu unterstützen. Zu diesem Zweck wird das Bewertungskriterium V6 definiert, das sich mit der Identifizierung von Inkonsistenzen in Zielmodellen beschäftigt. Es findet eine Abgrenzung der drei in Tabelle 3.10 vorgestellten Erfüllungsgrade statt. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Inkonsistenzen in Zielmodellen zu identifizieren. Das

Kriterium ist teilweise erfüllt, wenn der Ansatz es zwar nicht erlaubt, Inkonsistenzen in Zielmodellen zu identifizieren, aber dennoch einen Beitrag zur Identifikation von Inkonsistenzen in Zielmodellen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz es nicht erlaubt, Inkonsistenzen in Zielmodellen zu identifizieren, und auch keinen Beitrag zur Identifikation von Inkonsistenzen in Zielmodellen leisten kann.

Tabelle 3.10: Bewertungskriterium V6

erfüllt	Der Ansatz erlaubt es, Inkonsistenzen im Zielmodell zu erkennen und dadurch die Validierung des Zielmodells zu unterstützen.
teilweise erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen im Zielmodell zu erkennen, kann jedoch einen Beitrag zur Erkennung von Inkonsistenzen im Zielmodell leisten.
nicht erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen im Zielmodell zu erkennen.

Als weitere Eigenschaft eines Ansatzes, um zur Zielsetzung der Arbeit beizutragen, ist die Fähigkeit zu nennen, Inkonsistenzen im SoS-Modell zu identifizieren. Inkonsistenzen im SoS-Modell deuten auch auf abhängige Defekte im Zielmodell hin. Bspw. können Ziele identifiziert werden, die ausschließlich für ungültige SoS-Konfigurationen spezifiziert wurden. Dafür wird das Bewertungskriterium V7 definiert, das sich mit der Identifizierung von Inkonsistenzen in SoS-Modellen beschäftigt. Es erfolgt eine Unterscheidung der drei in Tabelle 3.11 vorgestellten Erfüllungsgrade. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Inkonsistenzen in SoS-Modellen zu identifizieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz es zwar nicht erlaubt, Inkonsistenzen in SoS-Modellen zu identifizieren, aber dennoch einen Beitrag zur Identifikation von Inkonsistenzen in SoS-Modellen leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz es nicht erlaubt, Inkonsistenzen in SoS-Modellen zu identifizieren, und auch keinen Beitrag zur Identifikation von Inkonsistenzen in SoS-Modellen leisten kann.

Tabelle 3.11: Bewertungskriterium V7

erfüllt	Der Ansatz erlaubt es Inkonsistenzen im SoS-Modell zu erkennen und dadurch die Validierung des SoS-Modell unterstützen.
teilweise erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen im SoS-Modell zu erkennen, kann jedoch einen Beitrag zur Erkennung von Inkonsistenzen im SoS-Modell leisten.
nicht erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen im SoS-Modell zu erkennen.

Zu den weiteren wichtigen Eigenschaften eines Ansatzes, um die Zielsetzung dieser Arbeit zu realisieren, gehört die Fähigkeit, Inkonsistenzen zwischen Ziel- und SoS-Modell zu

identifizieren, da diese auf Defekte in der Zielspezifikation hindeuten können. Dafür wird das Bewertungskriterium V8 definiert, das sich mit der Identifizierung von Inkonsistenzen zwischen Ziel- und SoS-Modell beschäftigt. Es werden die drei in Tabelle 3.12 vorgestellten Erfüllungsgrade unterschieden. Das Kriterium ist erfüllt, wenn der Ansatz es erlaubt, Inkonsistenzen zwischen Ziel- und SoS-Modell zu identifizieren. Das Kriterium ist teilweise erfüllt, wenn der Ansatz es zwar nicht erlaubt, Inkonsistenzen zwischen Ziel- und SoS-Modell zu identifizieren, aber dennoch einen Beitrag zur Identifikation von Inkonsistenzen zwischen Ziel- und SoS-Modell leisten kann. Das Kriterium ist nicht erfüllt, wenn der Ansatz es nicht erlaubt, Inkonsistenzen zwischen Ziel- und SoS-Modell zu identifizieren, und auch keinen Beitrag zur Identifikation von Inkonsistenzen zwischen Ziel- und SoS-Modell leisten kann.

Tabelle 3.12: Bewertungskriterium V8

erfüllt	Der Ansatz erlaubt es, Inkonsistenzen zwischen Ziel- und SoS-Modell zu erkennen und dadurch die Validierung des Zielmodells unter Berücksichtigung der verschiedenen SoS-Konfiguration zu unterstützen.
teilweise erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen zwischen Ziel- und SoS-Modell zu erkennen, kann jedoch einen Beitrag zur Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell zu erkennen leisten.
nicht erfüllt	Der Ansatz erlaubt es nicht, Inkonsistenzen zwischen Ziel- und SoS-Modell zu erkennen.

3.3 Ansätze zur Spezifikation von Zielen für SoS-Konfigurationen

Dieser Abschnitt stellt Ansätze zur Spezifikation von Zielen für SoS-Konfigurationen vor. Dabei wird zwischen Ansätzen zur Zielmodellierung für Systems-of-Systems und Ansätzen zur Zielmodellierung für variable Systeme unterschieden.

3.3.1 Ansätze zur Zielmodellierung für Systems-of-Systems

Im Rahmen der Zielmodellierung für Systems-of Systems steht die Unterscheidung der Ziele des betrachteten System-of-Systems von den Zielen der einzelnen konstituierenden Systeme im Fokus (Lewis et al. 2009). Diese beiden Ebenen (die manchmal auch als Makroebene und Mikroebene bezeichnet werden (Kopetz et al. 2016)) der Zielmodellierung für Systems-of-Systems ermöglichen die Analyse der Zusammenarbeit zwischen einzelnen Systemen, indem sie sich darauf konzentrieren, wie ihre individuellen Ziele zu den Zielen auf Systems-of-Systems-Ebene beitragen. Zusätzlich schlagen Cavalcante et al. (2015) eine neue Art von

Verknüpfung vor: Interaktionsverknüpfungen, um explizit emergentes Verhalten durch Ziele zu berücksichtigen, deren Erfüllung aus Interaktionen zwischen einzelnen Systemen resultiert. Neben solchen konzeptionellen Ansätzen gibt es auch spezifische Richtlinien und Sprachen für die Modellierung von System-of-Systems-Zielen und konstituierenden Systemzielen. Lewis et al. (2009) schlagen vor, getrennte UND/ODER-Zielbäume für die einzelnen Systeme und das System-of-Systems zu erstellen, um gemeinsame Ziele in den verschiedenen Zielmodellen der einzelnen Systeme sowie Zielkonflikte sowohl zwischen den einzelnen Systemen als auch den Gesamtzielen des System-of-Systems zu identifizieren. Garro und Tundis (2015) verwenden Stereotype, um die Ziele von Stakeholdern und von komplexen Systems-of-Systems zu charakterisieren, die zur Erreichung dieser Ziele herangezogen werden. In einem engen Zusammenhang mit den System-of-Systems-Zielen steht nach Silva et al. (2014) das Missionskonzept. Ziele sind mit der Mission des gesamten System-of-Systems und der Mission der konstituierenden Systeme verbunden. Diejenigen Ziele, die in Verbindung mit der Mission eines System-of-Systems stehen, werden durch die Zusammenarbeit der einzelnen Systeme erreicht. Daher unterbreiten Silva et al. (2015) den Vorschlag eines missionszentrierten System-of-Systems-Entwurfsprozesses, der eine dezidierte Missionsebene umfasst, in der die Missionen der einzelnen Systeme und des System-of-Systems modelliert werden. Für die Modellierung von Missionen in einem System-of-Systems-Kontext regen sie den mKAOS-Ansatz an (Silva et al. 2015; Silva und Batista 2018), der auf der KAOS-Zielmodellierungssprache (van Lamsweerde 2009) aufbaut und System-of-Systems-relevante Erweiterungen enthält. Zur Operationalisierung von Zielen weist mKAOS zwei Arten von Fähigkeitsmodellen auf, von denen sich eines mit der Modellierung des Informationsaustausches zwischen einzelnen Systemen und den daraus resultierenden Fähigkeiten des System-of-Systems befasst (bezeichnet als "kommunikative Fähigkeiten"). Darüber hinaus enthält mKAOS ein dezidiertes Modell für emergentes Verhalten, das solche System-of-Systems-Fähigkeiten gruppiert und mit den daraus resultierenden emergenten Eigenschaften/Funktionalitäten in Beziehung setzt. Garcés und Nakagawa (2017) liefern Richtlinien und Empfehlungen für die Erstellung von mKAOS-Modellen. Diese beinhalten auch globale Missionen eines System-of-Systems auf mehreren Abstraktionsebenen durch Zielverfeinerung und Abstraktion, um Rationalitäten hinter den Missionen eines System-of-Systems zu identifizieren.

Ansätze zur Zielmodellierung für Systems-of-Systems erlauben es, Ziele zu spezifizieren. (Kriterium S1 erfüllt.) Ansätze zur Zielmodellierung für Systems-of-Systems fokussieren üblicherweise jedoch nicht auf die Variabilität des System-of-Systems, sondern auf das Zusammenspiel zwischen dem System-of-Systems und den Einzelsystemen. Daher gilt S2 als nicht erfüllt. Die Zusammenhänge zwischen Zielen und SoS-Konfigurationen werden dabei nicht berücksichtigt. Daher gilt S3 als nicht erfüllt. Des Weiteren bieten Zielmodellierungsansätze für Systems-of-Systems keine Unterstützung bei der Reduktion des Umfangs. Folglich ist S4 ebenfalls nicht erfüllt. Tabelle 3.13 fasst die Bewertung für Ansätze zur Zielmodellierung für Systems-of-Systems zusammen.

Tabelle 3.13: Bewertung – Zielmodellierungsansätze für System-of-Systems

Kriterium		Bewertung
S1	Berücksichtigung von Zielen bei der Spezifikation	erfüllt
S2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen	nicht erfüllt
S3	Berücksichtigung der Relation von Zielen und SoS-Konfigurationen	nicht erfüllt
S4	Reduktion des Umfangs der zu erstellenden Spezifikation	nicht erfüllt

3.3.2 Ansätze zur Zielmodellierung für variable Systeme

Für die modellbasierte Spezifikation von Zielen von variablen Systemen können der Literatur drei Arten von Ansätzen entnommen werden.

- Ansätze, die existierende Zielmodellierungssprachen erweitern, um Variabilität im Zielmodell zu spezifizieren
- Ansätze, die die Verknüpfung von Zielmodellen mit Variabilitätsmodellen empfehlen
- Ansätze, die die Spezifikation von Zielen für variable Systeme in getrennten Modellen empfehlen

3.3.2.1 Ansätze zur Dokumentation von Variabilität in Zielmodellen

Mehrere Ansätze regen an, Zielmodelle zur Spezifikation von Variabilität zu verwenden. Bidian und Yu (2007) stellen zum Beispiel einen Ansatz zur Spezifikation von Variabilität in Zielmodellen vor, der den Zielmodellen Entscheidungsgrenzen hinzufügt. Entscheidungsgrenzen trennen die variablen Teile des Zielmodells von den nicht variablen Teilen. Andere Ansätze verwenden vorhandene Zielmodellelemente zur Spezifikation von Variabilität. González-Baixauli und Laguna (2007) empfehlen zum Beispiel die Verwendung von ODER-Dekompositions- und Contribution Links zur Spezifikation von Produktlinienvariabilität in Zielmodellen. Neben der Verwendung von Zielmodellen zur Spezifikation von Produktlinienvariabilität werden Zielmodelle auch zur Unterstützung der Produktanpassung verwendet. Liaskos et al. schlagen die Verwendung von Zielmodellen zur Erfassung der Variabilität in der frühen Anforderungsphase vor. In Liaskos et al. (2007) erörtern die Autoren die Rolle der Variabilität beim Design von Geschäftsprozessen und autonomen Systemen sowie bei der Informationsmodellierung und dem Datenbankdesign und zeigen auf, wie Zielmodelle zur Erfassung der Variabilität eingesetzt werden können. In Liaskos et al. (2009) und Liaskos et al. (2010) wird eine Erweiterung für Zielmodellierungssprachen vorgestellt, die die Spezifikation optionaler Ziele ermöglicht, was wiederum Analysen von Kompromissen zwischen Benutzerpräferenzen zulässt. Ein diesbezügliches Tool wird in Liaskos und Rogozhkin (2011) vorgestellt. In Liaskos et al. (2011ba 2012) führen die Autoren eine zusätzliche Erweiterung

ihres Ansatzes ein, um Mappings auf Einheiten von Quellcodes einzubeziehen, sodass Anforderungspräferenzen direkt zu Systemanpassungen führen können. In Santos et al. (2010) wird ein Ansatz zur Verwendung von Zielmodellen zur Spezifikation von Variabilität von BPMN-Prozessmodellen vorgestellt.

Die meisten Zielmodellierungssprachen bieten zwar eine gewisse Unterstützung bei der Spezifikation von Variabilität, wie z. B. ODER-Dekompositionen, aber sie bieten in der Regel keine Möglichkeit, optionale Ziele oder Kardinalitäten zu definieren. Um diese Unzulänglichkeiten zu beheben, wurden verschiedene Erweiterungen zu bestehenden Zielmodellierungssprachen vorgeschlagen. Silva et al. (2008) präsentieren zum Beispiel einen aspektorientierten i*-Ansatz zur Modellierung gemeinsamer und variabler Features einer Produktlinie mithilfe eines Zielmodells. Variable Ziele, Aufgaben, Ressourcen usw. werden den Aspekten entsprechend demjenigen Feature-Modell zugeordnet, das die Produktlinie beschreibt. Borba und Silva (2009) führen eine Erweiterung des i*-Ansatzes ein, die sieben verschiedene Arten von Means-End-Links unterscheidet, um mit dem Mangel an Kardinalitäten umzugehen, der zur Spezifikation von Feature-Gruppen mit Kardinalitäten erforderlich ist. Lapouchnian und Mylopoulos (2009) regen hingegen die Verwendung von Tags an. Elemente im Zielmodell werden mit Tags annotiert, die definieren, in welchem Kontext ein Element gültig ist, sodass ein parametrisiertes Zielmodell für verschiedene Kontexte entsteht. Kontextparameter können in Hierarchien angeordnet werden. Ist einem Ziel ein Unterparameter zugeordnet, so trifft dies auch auf die Oberparameter zu. In Lapouchnian und Mylopoulos (2011) wird der Ansatz auf die i*-Modellierungssprache angewandt, wobei sowohl SR- als auch SD-Diagramme berücksichtigt werden.

Die Tropos-Modellierungssprache wurde von Penserini et al. (2007) erweitert, um den Entwurf hochvariabler Systeme durch die explizite Modellierung von Alternativen zu unterstützen. Ali et al. (2009b) präsentieren einen Zielmodellierungsansatz für Systeme mit variablem Kontext, bei dem das Tropos-Zielmodell mit Fakten annotiert wird, die bestimmen, wann ein Element aus dem Zielmodell gültig ist. Fakten werden zu Formeln kombiniert, die relevante Kontexte beschreiben (z. B. Patient ist zu Hause und es regnet).

Zur Spezifikation von Variabilität mithilfe der GRL erweitern Liu et al. (2014a) die GRL um Feature-Modell-Konzepte. Darüber hinaus erweitern sie den GRL-Evaluationsalgorithmus, sodass er zur Ableitung gültiger Produktkonfigurationen verwendet werden kann, ohne dass ein separates Feature-Modell erstellt werden muss. Ein derartiges Tool wird in Liu et al. (2014b) vorgestellt.

Die KAOS-Methode wurde in Brunet et al. (2008); Semmak et al. (2008 2009) erweitert, um die Spezifikation von DomänenZielmodellen zu unterstützen. Ansätze zur Spezifikation von Variabilität in Map-Ziel-Modellen wurden von Bennasri und Souveyet (2004) vorgeschlagen, die einen Ansatz zur Spezifikation von Variabilität in Anforderungen vorstellen, der alternative Pfade in Map-Ziel-Modellen berücksichtigt, sowie von Rohleder (2008), der einen Visualisierungsansatz zur Darstellung von Variabilität in nicht funktionalen Anforderungen

auf der Basis von Map-Ziel-Modellen anregt. Shamsaei et al. (2013) verwenden Stereotype, um Zielmodelle zu annotieren, die Vorschriften beschreiben, die nur in bestimmten Situationen Gültigkeit besitzen. Ihr Ansatz erlaubt es, bei der Bewertung des Zielmodells für eine bestimmte Situation nur die relevanten Elemente zu berücksichtigen. Zur Modellierung von Softwareproduktlinien stellen Mussbacher (2008) eine aspektorientierte Erweiterung der User-Requirements-Notation vor. Diese ermöglicht die Modellierung der Ziele von Softwareproduktlinien auf aspektorientierte Weise. Die Features der Softwareproduktlinie werden als GRL-Graph modelliert. Neben einem Basis-Zielmodell gibt es Pointcut und Advice-Graphen, um die Variabilität der Softwareproduktlinie darzustellen.

Ansätze zur Dokumentation von Variabiltät in Zielmodellen erlauben es, Ziele zu spezifizieren. (Kriterium S1 erfüllt.) Die Dokumentation von Variabilität in einem Zielmodell erlaubt es, unterschiedliche SoS-Konfigurationen zu berücksichtigen. Daher gilt S2 als erfüllt. Die Zusammenhänge zwischen Zielen und SoS-Konfigurationen werden dabei ebenfalls berücksichtigt. Daher gilt S3 als erfüllt. Durch die Dokumentation von Variabilität im Zielmodell wird der Umfang der Spezifikation reduziert. Daher ist S4 erfüllt. Tabelle 3.14 fasst die Bewertung für Ansätze zur Dokumentation von Variabilität in Zielmodellen zusammen.

Tabelle 3.14: Bewertung – Ansätze zur Dokumentation von Variabilität in Zielmodellen

Kriterium		Bewertung
S1	Berücksichtigung von Zielen bei der Spezifikation	erfüllt
S2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen	erfüllt
S3	Berücksichtigung der Relation von Zielen und SoS-Konfigurationen	erfüllt
S4	Reduktion des Umfangs der zu erstellenden Spezifikation	erfüllt

3.3.2.2 Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen

Die Verknüpfung von Ziel- und Variabilitätsmodellen wurde von mehreren Autoren vorgeschlagen. Park et al. (2004) präsentieren einen Ansatz zur Erhebung von Anforderungen an Software-Produktlinien mittels ziel- und szenariobasiertem Requirements Engineering. Ziele sind zu Szenarien und die Szenarien zu Features der Produktlinie relationiert, wodurch eine indirekte Verbindung zwischen Zielen und Features hergestellt wird. Lee et al. (2014) stellen einen Prozess zur Erstellung von Feature-Modellen für Produktlinien vor, der auf Viewpoints basiert, die den Lösungsraum vom Problemraum trennen. Morandini et al. (2017) führen ein Rahmenwerk für Anforderungen für adaptive Softwaresysteme ein, das zielorientierte Konzepte und hochvariable Entwurfsmodelle verknüpft. Das Anforderungsmodell lässt sich auf Software-Prototypen mit einer agentenorientierten Architektur abbilden. Asadi et al. (2011) präsentieren einen Ansatz zur Erstellung von Feature-Modellen auf der Basis von i*-Zielmodellen. Ziel- und Feature-Modelle werden durch die Annotation von Features mit

booleschen Variablen verknüpft, die den Zielen aus dem Zielmodell entsprechen. Auf diese Weise können die Ziele als Begründung für die Featureauswahl herangezogen werden. Mehrere Ansätze regen die Verwendung von dezidierten Modellen für die Verknüpfung von Zielen und variablen Features an. Jarzabek et al. (2006) präsentieren einen Ansatz zur Erstellung eines Feature-Softgoal-Interdependency-Graphen, der Zielmodelle mit Feature-Modellen verknüpft. Der Ansatz zielt darauf ab, die Berücksichtigung von Qualitätsmerkmalen in der Domänenanalyse für Produktlinien zu unterstützen. Wang et al. (2011 2012) präsentieren einen Ansatz zur Erhöhung der Erfüllung von Softgoals für eine kontextsensitive adaptive Anwendung, die ebenfalls Softgoal-Interdependency-Graphen verwendet. Mussbacher et al. (2012) stellen einen aspektorientierten Ansatz für das Produktlinien-Engineering vor. GRL-Modelle und Feature-Modelle werden durch Feature-Impact-Modelle verknüpft. Ein Feature-Impact-Modell zeigt ein Feature und die Ziele, auf die es sich bezieht. Semmak und Brunet (2006) schlagen die Verwendung von Wissensfragmenten vor, die ein Ziel, eine Geschäftsregel und ein konzeptuelles Fragment verbinden, das Abhängigkeiten zwischen Elementen der Domäne ausdrückt, die zur Erreichung eines Ziels interagieren. Kim et al. (2004) und befassen sich mit der Verwendung von variablen Use-Case-Modellen, um die Produktlinienvariabilität mit Zielen und Szenarien zu verknüpfen. Duran et al. (2015) greifen einen Ansatz auf, der die Wiederverwendung von Zielmodellen unterstützt. Der Ansatz erlaubt die Verwendung von relativen Contributions und die Definition von Constraints zwischen Tasks mithilfe eines Feature-Modells.

Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen erlauben es, Ziele zu spezifizieren. (Kriterium S1 erfüllt.) Die Verknüpfung von Ziel- und Variabilitätsmodellen erlaubt die Berücksichtigung unterschiedlicher SoS-Konfigurationen. Daher gilt S2 als erfüllt. Die Zusammenhänge zwischen Zielen und SoS-Konfigurationen werden dabei ebenfalls berücksichtigt. Daher gilt S3 als erfüllt. Durch die Verknüpfung von Ziel- und Variabilitätsmodellen wird der Umfang der Spezifikation reduziert. Daher ist S4 erfüllt. Tabelle 3.15 fasst die Bewertung von Ansätzen zur Dokumentation von Variabilität in Zielmodellen zusammen.

Tabelle 3.15: Bewertung – Ansätze zur Verknüpfung von Ziel- und Variabiltätsmodellen

Kriterium		Bewertung
S1	Berücksichtigung von Zielen bei der Spezifikation	erfüllt
S2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen	erfüllt
S3	Berücksichtigung der Relation von Zielen und SoS-Konfigurationen	erfüllt
S4	Reduktion des Umfangs der zu erstellenden Spezifikation	erfüllt

Die Lösungsidee basiert auf der Verknüpfung von Ziel- und Variabilitätsmodellen, um die Zusammenhänge zwischen Zielen und SoS-Konfigurationen zu spezifizieren. Diese Vorgehens-

weise erlaubt es auch, für Systems-of-Systems mit vielen SoS-Konfigurationen Ziele für alle SoS-Konfigurationen in einer handhabbaren Weise zu spezifizieren.

3.3.2.3 Ansatz von Goldsby et al. zur Dokumentation von Variabilität in getrennten Zielmodellen

Während die meisten Ansätze zur Spezifikation von Variabilität verschiedene Varianten in ein Zielmodell integrieren, schlägt der Ansatz von Goldsby et al. (2008) zur Entwicklung adaptiver Systeme vor, die verschiedenen Ziele in separaten Zielmodellen zu spezifizieren, d. h. ein Zielmodell für Situation A, ein Zielmodell für Situation B etc.

Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen gestatten die Spezifikation von Zielen. (Kriterium S1 erfüllt.) Die Dokumentation von Variabilität in getrennten Zielmodellen erlaubt es, unterschiedliche SoS-Konfigurationen zu berücksichtigen. Daher gilt S2 als erfüllt. Die Zusammenhänge zwischen Zielen und SoS-Konfigurationen werden dabei ebenfalls berücksichtigt. Daher gilt S3 als erfüllt. Durch die Dokumentation von Variabilität in getrennten Zielmodellen wird der Umfang der Spezifikation nicht reduziert. Daher ist S4 nicht erfüllt. Tabelle 3.16 fasst die Bewertung von Ansätzen zur Dokumentation von Variabilität in getrennten Zielmodellen zusammen.

Tabelle 3.16: Bewertung – Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen

Kriterium		Bewertung
S1	Berücksichtigung von Zielen bei der Spezifikation	erfüllt
S2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen	erfüllt
S3	Berücksichtigung der Relation von Zielen und SoS-Konfigurationen	erfüllt
S4	Reduktion des Umfangs der zu erstellenden Spezifikation	nicht erfüllt

3.4 Ansätze zur Unterstützung der Validierung von Zielmodellen für SoS-Konfigurationen

Dieser Abschnitt befasst sich mit Ansätzen, die die Validierung von Zielmodellen für SoS-Konfigurationen unterstützen können. Betrachtet werden Validierungsansätze und Analyseansätze für Zielmodelle, Featuremodelle, Zielmodelle für variable Systeme sowie Produktlininen- und Software-Variabilität.

3.4.1 Validierungsansätze

Dieser Abschnitt stellt Validierungsansätze vor, die speziell für die Validierung von Zielmodellen oder Entwicklungsartefakten variabler Systeme entwickelt wurden.

Tabelle 3.17: Bewertung – Ansätze zur Validierung von Zielmodellen

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	nicht erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	teilweise erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	nicht erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	nicht erfüllt

3.4.1.1 Ansätze von Uchitel et al. und Hassine und Amyot zur Validierung von Zielmodellen

Es existieren einige wenige Ansätze, die speziell für die Validierung von Zielmodellen entwickelt wurden. Uchitel et al. (2006) stellen einen Ansatz vor, der es erlaubt, aus Zielmodellen und dazugehörigen Szenarien das Systemverhalten zu animieren. Diese Animation kann dann von Stakeholdern bewertet werden, sodass Defekte im Zielmodell sowie in den Szenarien zu erkennen sind. Hassine und Amyot (2016) empfehlen, die Validierung von Zielmodellen durch Umfragen zu unterstützen. Dabei wird aus dem Zielmodell ein Fragebogen generiert, der den Stakeholdern die Gelegenheit gibt, ihre Zustimmung zu den modellierten Sachverhalten, die im Rahmen des Fragebogens natürlichsprachlich beschrieben sind, auf einer 7-Punkt-Likert-Skala anzugeben. Die Ergebnisse des Fragebogens werden statistisch ausgewertet. Hierdurch können Defekte im Zielmodell und Konflikte zwischen Stakeholdern identifiziert und Vorschläge für die Konfliktlösung unterbreitet werden (Hassine und Amyot 2017).

Ansätze zur Validierung von Zielmodellen unterstützen die Validierung von Zielmodellen (Kriterium V1 erfüllt). Des Weiteren können diese Ansätze auch einen Beitrag zur Erkennung von Inkonsistenzen in Zielmodellen leisten, da solche Inkonsistenzen im Rahmen der Validierung auffallen können. Jedoch ist nicht sichergestellt, dass alle Inkonsistenzen erkannt werden. Aus diesem Grund gilt das Kriterium V6 als teilweise erfüllt. Die weiteren Kriterien werden von Ansätzen zur Validierung von Zielmodellen nicht erfüllt. Tabelle 3.17 fasst die Bewertung von Ansätzen zur Validierung von Zielmodellen zusammen.

3.4.1.2 Ansätze von Kim et al. und de Mello et al. zur Validierung von Entwicklungsartefakten für variablen Systeme

Um die Validierung von Entwicklungsartefakten für variable Systeme zu unterstützen wurden in der Literatur Checklisten für die Inspektion solcher Entwicklungsartefakte vorgeschlagen. Kim et al. (2008) haben vier Checklisten für die Inspektion der Architektur einer Softwareproduktlinie entwickelt. Diese Checklisten berücksichtigen Gemeinsamkeiten und Variabilität, Architekturebenen, Abstraktionsmechanismen und Schnittstellen. de Mello et al. (2014)

Tabelle 3.18: Bewertung – Ansätze zur Validierung von Entwicklungsartefakten für variable Systeme

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	nicht erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	nicht erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	teilweise erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	nicht erfüllt

haben eine Checkliste für die Inspektion von Feature-Modellenentwicklt. Diese umfasst insgesamt 24 Punkte die sich in drei Bereiche aufteilen: einzelne Feature, Beziehungen zwischen Featuren und Kompositionsregeln.

Ansätze zur Validierung von von Entwicklungsartefakten für variablen Systeme können SoS-Konfigurationen bei der Validierung berücksichtigen. (Kriterium V2 erfüllt.) Des Weiteren können diese Ansätze auch einen Beitrag zur Erkennung von Inkonsistenzen in SoS-Modellen liefern, da solche Inkonsistenzen im Rahmen der Validierung auffallen können. Jedoch ist nicht sichergestellt, dass alle Inkonsistenzen erkannt werden. Aus diesem Grund gilt das Kriterium V7 als teilweise erfüllt. Die weiteren Kriterien werden von Ansätzen zur Validierung von von Entwicklungsartefakten für variablen Systeme nicht erfüllt. Tabelle 3.18 fasst die Bewertung für Ansätze zur Validierung von Entwicklungsartefakten für variablen Systeme zusammen.

3.4.2 Ansätze zur Analyse von Zielmodellen

In der Literatur werden verschiedene Arten von Ansätzen angeführt, die genutzt werden, um Zielmodelle zu analysieren (Horkoff und Yu 2011). Es können die folgenden Kategorien abgegrenzt werden:

- Erfüllbarkeitsanalyseansätze: befassen sich mit der Frage, welche Ziele erfüllbar sind, falls bestimmte Ziele erfüllt sind.
- Berechnung von Metriken: ermitteln anhand des Zielmodells Kennwerte, aus denen sich Eigenschaften des Systems ableiten lassen.
- Planungsansätze: ermitteln, welche Möglichkeiten es gibt, um Ziele zu erfüllen, und versuchen, auf Basis von vordefinierten Kriterien die beste Möglichkeit zu finden.
- Simulationsansätze: erweitern herkömmliche Zielmodellierungssprachen um zeitliche Informationen wie Vor- und Nachbedingungen, was es erlaubt, das Verhalten von Akteuren zu simulieren.

- Modelcheckingansätze: erlauben es, Zielmodelle auf bestimmte Eigenschaften, wie bspw. Konsistenz, zu prüfen.

Bezogen auf die Problemstellung dieser Arbeit, die sich mit der Spezifikation, Validierung und Erfüllbarkeitsprüfung der Ziele von Systems-of-Systems beschäftigt, sind Erfüllbarkeitsanalysen und Modelcheckingansätze von Interesse, da diese sich eignen, um die Validierung von Zielmodellen zu unterstützen.

3.4.2.1 Ansätze zur Analyse der Erfüllbarkeit für Zielmodelle

Erfüllbarkeitsanalysen lassen sich in Vorwärts- und Rückwärtsanalysen unterteilen (Horkoff und Yu 2013). Vorwärtsanalysen (bspw. (Amyot et al. 2010; Chung et al. 2000; Giorgini et al. 2002 2005; Sebastiani et al. 2004; Horkoff und Yu 2009; Maiden et al. 2007)) beschäftigen sich damit, welche Auswirkungen erfüllbare Ziele auf die Erfüllbarkeit anderer Zielen haben. Rückwärtsanalysen (bspw. (Giorgini et al. 2005; Sebastiani et al. 2004; Horkoff und Yu 2010; Letier und van Lamsweerde 2004)) setzen sich mit der Frage auseinander, welche Ziele erfüllbar sein müssen, damit ausgewählte Ziele erfüllbar sind.

Ein weiteres Unterscheidungsmerkmal für Erfüllbarkeitsanalysen ist die Genauigkeit, mit der Erfüllbarkeitsgrade angegeben werden können. Manche Ansätze (bspw. (Maiden et al. 2007)) unterscheiden nur zwischen erfüllbar und nicht erfüllbar. Andere Ansätze (bspw. (Amyot et al. 2010; Chung et al. 2000)) erlauben differenziertere Unterscheidungen, wie teilweise erfüllbar, teilweise nicht erfüllbar. Neben diesen qualitativen Ansätzen gibt es Ansätze für eine Erfüllbarkeitsanalyse, die mit quantitativen Werten arbeiten, um Wahrscheinlichkeiten für Erfüllbarkeit auszudrücken.

Eine Herausforderung bei qualitativen Erfüllbarkeitsanalyseansätzen ist der Umgang mit widersprüchlichen Informationen, wenn bspw. ein Ziel einen positiven Einfluss auf die Erfüllbarkeit eines Ziels ausübt, ein anderes Ziel aber negativen Einfluss auf die Erfüllbarkeit des gleichen Ziels nimmt. Für diese Problematik lassen sich in der Literatur drei unterschiedliche Arten von Lösungen finden: der Konflikt wird nicht aufgelöst (Giorgini et al. 2002 2005; Sebastiani et al. 2004), der Konflikt wird durch vorher definierte Regeln aufgelöst (Amyot et al. 2010), der Konflikt muss durch den Entwickler aufgelöst werden (Chung et al. 2000; Horkoff und Yu 2009 2010).

Der Ansatz von Grubb und Chechik (2021) erlaubt es, den Einfluss der Zeit auf die Erfüllbarkeit von Zielen zu spezifizieren und bei der Analyse zu berücksichtigen. Dazu werden für die Ziele Funktionen spezifiziert, die angeben, welchen Erfüllbarkeitsgrad ein Ziel zu welchem Zeitpunkt aufweist.

Ansätze zur Erfüllbarkeitsanalyse unterstützen die Validierung von Zielmodellen. Daher gilt Kriterium V1 als erfüllt. Des Weiteren unterstützen Ansätze zur Erfüllbarkeitsanalyse die Erkennung von Inkonsistenzen in Zielmodellen. Aus diesem Grund gilt das Kriterium V6 als

erfüllt. Die weiteren Kriterien werden von Ansätzen zur Validierung von Zielmodellen nicht erfüllt. Tabelle 3.19 fasst die Bewertung von Ansätzen zur Erfüllbarkeitsanalyse zusammen.

Tabelle 3.19: Bewertung – Erfüllbarkeitsanalyseansätze

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	nicht erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	nicht erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	nicht erfüllt

Die Lösungsidee nutzt Erfüllbarkeitsanalyseansätze, um Inkonsistenzen in Zielmodellen zu identifizieren. Dies unterstützt die Validierung, da Defekte automatisiert erkannt werden können.

3.4.2.2 Ansätze zum Modelchecking von Zielmodellen

Modelcheckingansätze prüfen, ob bestimmte Eigenschaften für ein Verhaltensmodell gelten (Clarke et al. 1999). Da Zielmodelle kein Verhalten spezifizieren, sind Anpassungen bzw. Erweiterungen erforderlich, um Modelchecking durchzuführen.

Fuxman et al. (2000 2004) stellen einen Ansatz vor, um iStar SD-Diagramme zu prüfen. Dazu wird das Diagramm in Formal Tropos umgewandelt, das neben der Spezifikation von Akteuren und Abhängigkeiten auch dynamische Aspekte berücksichtigt. Das T-Tool kann dann die Formal-Tropos-Beschreibung in ein Eingabemodell für einen Modelchecker überführen, sodass das SD-Diagramm geprüft werden kann. Giorgini et al. (2004) befassen sich mit einer Erweiterung für Tropos, in der Security und Vertrauen (Trust) spezifiziert werden können. Dadurch können die Abhängigkeiten auf Konsistenz geprüft werden.

Deb und Chaki (2020) stellen einen Ansatz vor, um Zielmodelle für iStar-Modelle per Modelchecking zu prüfen, bei denen jedem Element eines SR-Diagramms drei Zustände zugeordnet werden: *Nicht erstellt*, *Erstellt nicht erfüllt* und *Erfüllt*. Dadurch lässt sich das iStar-Modell in ein Verhaltensmodell überführen, in dem dann temporale Properties geprüft werden können, wie bspw. ob ein bestimmtes Ziel immer nach der Erfüllung eines anderen Ziels erfolgt. Deb et al. (2016) führen ein Tool ein, das ein iStar-Modell in ein Verhaltensmodell überführen kann.

Abeywickrama und Zambonelli (2012) stellen einen Ansatz vor, um Zielmodelle für adaptive Syteme zu modelchecken. Dazu werden aus den Tasks des Zielmodells Verhaltensmodelle für servicebasierte Komponenten abgeleitet, die dann überprüft werden können.

Der Ansatz von Ogawa et al. (2008) erlaubt es, aus KAOS-Zielmodellen LTL-Formeln für das Modelchecking von Verhaltensmodellen abzuleiten. So wird es möglich, Verhaltensmodelle gegen die spezifizierten Ziele zu prüfen. Ein solcher Ansatz wäre bei einer späteren Prüfung der Verhaltenspezifikation gegen das Zielmodell zwar hilfreich, jedoch kann er nicht die im Vorfeld erforderliche Validierung des Zielmodells selbst unterstützen.

Ansätze zum Modelchecking von Zielmodellen unterstützen die Validierung von Zielmodellen. Daher gilt Kriterium V1 als erfüllt. Des Weiteren unterstützen Ansätze zum Modelchecking von Zielmodellen die Erkennung von Inkonsistenzen in Zielmodellen. Aus diesem Grund gilt das Kriterium V6 als erfüllt. Die weiteren Kriterien werden von Ansätzen zur Validierung von Zielmodellen nicht erfüllt. Tabelle 3.20 fasst die Bewertung von Ansätzen zum Modelchecking von Zielmodellen zusammen.

Tabelle 3.20: Bewertung – Ansätze zum Modelchecking von Zielmodellen

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	nicht erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	nicht erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	nicht erfüllt

3.4.3 Ansätze zur Analyse von Feature-Modellen

In der Literatur existiert eine Vielzahl von Ansätzen, um Feature-Modelle zu analysieren. Feature-Modelle werden insbesondere im Rahmen von Softwareproduktlinien häufig zur Spezifikation von Variabilität verwendet (Galindo et al. 2019). Auf Feature-Modellen lässt sich eine Vielzahl unterschiedlicher automatisierter Analysen durchführen. In Benavides et al. (2010) wird ein Ansatz vorgestellt, mit dem sich ermitteln lässt, ob ein Feature-Modell ein valides Feature-Modell ist (d. h., dass mindestens ein Produkt aus dem Feature-Modell abgeleitet werden kann). Des Weiteren existieren Ansätze, um zu prüfen, ob ein bestimmtes Produkt aus einem Feature-Modell ableitbar ist (Batory 2005; Kang et al. 1990; Trinidad et al. 2008) oder ob ein bestimmtes Teilprodukt ableitbar ist (Benavides et al. 2010). Weitere Ansätze dienen dazu, alle durch ein Feature-Modell spezifizierten validen Produkte abzuleiten (Benavides et al. 2010) bzw. die Anzahl aller validen Produkte, die sich aus einem Feature-Modell ableiten lassen, zu bestimmen (Benavides et al. 2005; Czarnecki und Kim 2005; van Deursen und Klint 2004) oder alle Produkte zu bestimmen, die bestimmte Eigenschaften besitzen (Czarnecki und Kim 2005). Diverse Ansätze beschäftigen sich mit der Identifikation von Anomalien in Feature-Modellen, wie bspw. tote Features (d. h. Features, die in keinem

Produkt vorhanden sind) (Benavides et al. 2010), fälschlicherweise als optional spezifizierte Features, die in jedem Produkt vorhanden sind (Benavides et al. 2010), nicht instantiierbare Gruppenkardinalitäten (Trinidad et al. 2004) oder Redundanzen (Massen und Lichter 2004). Des Weiteren können für ein Feature-Modell die Kern-Features (d. h. die Features, die in allen Produkten vorhanden sind) und Varianten-Features (d. h. die Features, die nicht in allen Produkten vorhanden sind) bestimmt werden (Trinidad et al. 2004) sowie atomare Mengen an Features (d. h. Features, die immer gemeinsam vorhanden oder nicht vorhanden sind) (Segura 2008; Zhang et al. 2004). Solche Analysen unterstützen nicht nur die Konfiguration von Produkten, sondern auch die Auswahl von Testfällen, die Evolution von Produktlinien und das Reverse Engineering (Galindo et al. 2019). Ein weiterer häufiger Einsatzzweck von Feature-Analysen ist die Erkennung von Feature-Interactions, d. h. die Auswirkungen des Zusammenspiels von unterschiedlichen Features (Apel und Kästner 2009; Apel et al. 2011 2013). Than Tun et al. (2009) stellen einen auf dem Zave-Jackson Framework (Zave und Jackson 1997) basierenden Ansatz vor, um optimale Konfigurationen aus Anforderungen abzuleiten. Der Ansatz definiert drei Arten von Feature-Modellen, die miteinander verknüpft werden. Das Anforderungs-Feature-Modell definiert die Anforderungen, die die Produktlinie erfüllen soll. Das Problem-Welt-Feature-Modell beschreibt die Features der Kontexte, in denen das System möglicherweise eingesetzt wird. Das Spezifikations-Feature-Modell beschreibt die Software-Features des Systems. Durch die Formalisierung der Feature-Modelle und deren Beziehungen lässt sich ermitteln, ob eine optimale Konfiguration existiert, d. h. eine Konfiguration, die die Anforderungen erfüllt.

Ansätze zur Analyse von Feature-Modellen können unterschiedliche SoS-Konfigurationen bei der Validierung berücksichtigen. Daher gilt V2 als erfüllt. Des Weiteren unterstützen Ansätze zur Analyse von Feature-Modellen die Erkennung von Inkonsistenzen in SoS-Modellen. Aus diesem Grund gilt das Kriterium V als erfüllt. Die weiteren Kriterien werden von Ansätzen zur Validierung von Zielmodellen nicht erfüllt. Tabelle 3.21 fasst die Bewertung der Ansätze zur Analyse von Feature-Modellen zusammen.

Tabelle 3.21: Bewertung – Ansätze zur Analyse von Feature-Modellen

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	nicht erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	nicht erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	nicht erfüllt

Die Lösungsidee nutzt Ansätze zur Analyse von Feature-Modellen, um Inkonsistenzen in SoS-Modellen zu identifizieren. Dies unterstützt die Validierung, da Defekte automatisiert erkannt werden können.

3.4.4 Ansätze zur Analyse von Zielmodellen für variable Systeme

In der Literatur sind zwei Arten von Zielanalyseansätzen anzutreffen:

- Ansätze, bei denen ein Zielmodell genutzt wird, um ein System mit variablen Zusammensetzungen (bspw. das Produkt einer Produktlinie) zu konfigurieren, d. h. eine Systemzusammensetzung anhand von Kriterien zu bestimmen.

- Ansätze, die sich mit der zielorientierten Entwicklung von Systemen befassen, die sich an variable Umgebungen anpassen müssen.

3.4.4.1 Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen

Es existieren zielorientierte Produktkonfigurationsansätze, die in der Regel Zielmodelle als Mittel für die Stakeholder verwenden, um ihre Präferenzen auszudrücken, die bei Produktkonfigurationen berücksichtigt werden müssen. Gonzales-Baixauli et al. (2004) präsentieren einen Ansatz zur Auswahl der geeignetsten Variante auf der Grundlage der spezifizierten Softgoals. Zwei UND/ODER-Bäume werden manuell erstellt; einer enthält alle Ziele und Tasks des Systems, der andere alle Softgoals. Eine Variante wird durch einen ODER-Pfad im Ziel-/Taskbaum dargestellt. Die Ziele/Tasks werden manuell über Korrelations-Links mit Softgoals verknüpft. Es gibt fünf Arten von Korrelations-Links: Break (Wert 1-0,5), Hurt (0,5-0), Unknown (0), Help (0-0,5) und Make (0,5-1). Softgoals werden Prioritätswerte auf einer Perzentilskala zugewiesen. Auf der Grundlage dieser Parameter berechnet das Tool eine Bewertung für jede Variante, um die Softgoal-Erfüllung zu optimieren.

Der Ansatz von Hui et al. (2003) nutzt Zielmodelle zusammen mit Nutzerfähigkeiten und Nutzerpräferenzen, um variable Software an Nutzer anzupassen. Dieser Ansatz wird in Liaskos et al. (2005) um parametrisierbare Contribution-Links erweitert. Diese Parameter können sich auf der Grundlage von Werten ändern, die von den Benutzern ausgewählt werden (z. B. berechnet die Prüfung auf neue Nachrichten alle x Minuten verschiedene Beitragswerte zur Erhöhung der Verfügbarkeit und zur Verringerung der Netzwerknutzung). António et al. (2009) stellen einen Ansatz vor, der i*-Zielmodellierung und Feature-Modelle kombiniert, um die Entwicklung von Produktlinien zu unterstützen. Zunächst werden Domänen-Zielmodelle erstellt, daraufhin wird ein Feature-Modell der Produktlinie abgeleitet. Im Application Engineering werden Feature- und Zielmodelle konfiguriert, um Modelle für ein Produkt zu erstellen. Der Ansatz wird in Silva et al. (2011) um eine Erweiterung von i* ergänzt, diese beinhaltet

Kardinalitäten zur besseren Darstellung von Feature-Gruppen sowie einen Prozess zur Identifizierung von Features und zur Konfiguration von Feature-Modellen. Anders als in António et al. (2009) werden nur einige ausgewählte intentionale Elemente auf Features abgebildet und nur diesen werden Kardinalitäten zugewiesen. Guedes et al. (2011 2013) erweitern diesen Ansatz und beziehen die Spezifikation von Szenarien mithilfe von Use-Case-Templates ein.

Bagheri et al. (2010) präsentieren einen Ansatz zur Konfiguration von Software-Produktlinien auf der Grundlage von Stakeholder-Constraints. Die Stakeholder-Constraints werden in Aussagenlogik spezifiziert und das Feature-Modell wird dementsprechend konfiguriert. Soltani et al. (2012) verwenden Planungstechniken der Künstlichen Intelligenz, um Features einer Software-Produktlinie auszuwählen, die sowohl die Präferenzen der Stakeholder für funktionale und nicht funktionale Anforderungen als auch Constraints berücksichtigt.

Noorian et al. (2014) präsentieren einen Ansatz zur besseren Auswahl von Features von Produktlinien entsprechend den Bedürfnissen der Stakeholder. Zu diesem Zweck ordnet ihr Ansatz halbautomatisch Elemente aus Zielmodellen und Feature-Modellen zu. Der Ansatz erfordert die Existenz von Trace-Links zwischen dem Feature-Modell und natürlichsprachlichen Dokumenten (z. B. Interviewprotokollen) sowie zwischen dem Zielmodell und natürlichsprachlichen Dokumenten. Mithilfe von Natural-Language-Processing-Techniken werden die natürlichsprachlichen Dokumente automatisch analysiert und das Feature-Modell und das Zielmodell dementsprechend annotiert, die Zuordnung wird mittels Integer-Linear-Programmierung identifiziert. In Noorian et al. (2017) stellen die Autoren einen Prozess zum Finden der optimalen Konfiguration für ein Produkt einer Produktlinie auf der Grundlage der Bedürfnisse der Stakeholder vor. Das Feature-Modell, das Zielmodell und ihre Beziehungen werden in ein Integer-Linear-Programm transformiert, das dann verwendet wird, um die optimale Lösung zu finden. Der Ansatz von Asadi et al. (2014) beabsichtigt, die optimale Konfiguration auf der Grundlage nicht funktionaler Anforderungen zu finden. Er transformiert Feature-Modelle in einen hierarchischen Task-Netzwerk-Formalismus, der mit einem hierarchischen Task-Netzwerk-Planer analysiert werden kann, um die optimale Konfiguration basierend auf den Präferenzen der Stakeholder bezüglich nicht funktionaler Anforderungen festzustellen. Stein et al. (2014) präsentieren einen Ansatz zum Finden von Feature-Modell-Konfigurationen gemäß multiplen Stakeholder-Präferenzen, der Single-Winner-Voting-Strategien aus der Social-Choice-Theorie verwendet. Nakagawa et al. (2011) stellen einen Architektur-Compiler für selbst adaptive Systeme vor, der Architekturkonfigurationen auf der Grundlage zielorientierter Anforderungsbeschreibungen generiert.

Ansätze zur zielorientierten Konfiguration von Systemen unterstützen zu einem gewissen Grad die Validierung. Dies ist jedoch nicht ihr Hauptzweck. Daher gilt Kriterium V1 als teilweise erfüllt. Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen berücksichtigen die Variabilität, die durch unterschiedliche SoS-Konfigurationen entsteht. Daher gilt auch V2 als teilweise erfüllt. Im Rahmen der zielorientierten Konfiguration von Systemen können Inkonsistenzen im Zielmodell, im SoS-Modell

und zwischen dem Ziel- und SoS-Modell erkannt werden. Jedoch ist nicht sichergestellt, dass dies zuverlässig geschieht. Daher gelten V6, V7 und V8 ebenfalls als teilweise erfüllt. Tabelle 3.22 fasst die Bewertung der Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen zusammen.

Tabelle 3.22: Bewertung – Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	teilweise erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	teilweise erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	teilweise erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	teilweise erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	teilweise erfüllt

3.4.4.2 Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext

Darüber hinaus existieren Konfigurationsansätze für dynamische Software-Produktlinien (Bencomo et al. 2012; Gámez et al. 2015; Hallsteinsen et al. 2006 2008). Die Konfigurationen einer dynamischen Software-Produktlinie ändern sich, um das System an einen sich ändernden Kontext anzupassen (Gámez et al. 2015). Daher befassen sich viele Ansätze damit, die optimale Konfiguration für einen gegebenen Kontext zu finden. Zu diesem Zweck geht es bei den Ansätzen zur Konfiguration dynamischer Software-Produktlinien um die Überwachung des Kontextes und die Entscheidung, wann und wie sie sich anpassen.

Neben der Konfiguration von Systemen zur Optimierung der Zufriedenheit von Stakeholdern oder Softgoals gibt es Ansätze zur Konfiguration von Systemen, die sich an spezifische und sich häufig ändernde Rahmenbedingungen anpassen. Lapouchnian et al. (2006) schlagen zum Beispiel die Verwendung von Zielmodellen für autonome Computersysteme vor, die ihr Verhalten anpassen können. Manuell erstellte i*-Zielmodelle werden mit Vorbedingungen für die Zielausführung annotiert (z. B. drahtloses Netzwerk verfügbar) und Contribution-Links können parametrisiert werden, um die Berücksichtigung verschiedener Werte zu ermöglichen (z. B. Serverlast >300). In Lapouchnian et al. (2007) stellen die Autoren einen Ansatz zur Modellierung von Geschäftsprozessen mithilfe von Zielmodellen vor. Ein Zielmodell spezifiziert einen alternativen Prozess und ist annotiert, um parallele oder sequenzielle Ausführungen von Zielen/Aufgaben darzustellen. Das Zielmodell wird dann analysiert, um optimale Konfigurationen entsprechend den aktuellen Business- oder Kundenpräferenzen zu finden. Peng et al. (2012) führen einen control-theoretischen Self-Tuning-Ansatz ein, mit dem die Erfüllung von Qualitätsanforderungen in unsicheren und sich verändernden Umgebungen optimiert

werden kann, und Salehie et al. (2012) präsentieren einen Ansatz zur Handhabung variabler Sicherheits-Assets zur Laufzeit. Mithilfe eines Asset-Modells, eines Zielmodells und eines Threat-Modells wird ein Kausal-Netzwerk aufgebaut, das für die Analyse der Systemsicherheit in verschiedenen Situationen eingesetzt wird.

Ali et al. (2009a) schlagen die Integration von Zielmodellen, Feature-Modellen und Problem-Frames vor, um Konflikte um gemeinsam genutzte Ressourcen unter Berücksichtigung variabler Kontexte besser erkennen zu können. In Ali et al. (2010) entwickeln die Autoren einen Ansatz zur Identifizierung der Konfigurationen mit minimalen Entwicklungskosten, der alle Ziele in allen ausgewählten Kontexten erfüllt und in Ali et al. (2013) weiter ausgebaut wird, um Konsistenzprüfungen durchzuführen, um sicherzustellen, dass widersprüchliche Kontexte erkannt werden (z. B. Patient ist zu Hause und Patient besucht Nachbarn). Die Konfliktanalyse wird durchgeführt, um widersprüchliche Handlungen (z. B. gleichzeitiges Öffnen und Schließen von Fenstern) und Konflikte um Ressourcen (gleichzeitiges Rufen der Polizei und des Pflegepersonals per Telefon) zu erkennen. Asadi et al. (2016) präsentieren einen Ansatz, der GRL-Zielmodelle und Feature-Modelle und ihre Zusammenhänge in Beschreibungslogik überführt und auch auf Inkonsistenzen überprüft. Der in Anda und Amyot (2019) vorgestellte Ansatz transformiert Ziel- und Feature-Modelle in arithmetische Semantik, um die Auswahl optimaler Adaptionsstrategien zu ermöglichen.

Guedes et al. (2017) erweitern den Ansatz aus Silva et al. (2011) auf dynamische Software-Produktlinien. Der mögliche Kontext, dem eine dynamische Software-Produktlinie unterworfen sein kann (z. B. schwache Batterie, kein Netzwerk), führt zu Rekonfigurationen der Produktlinie, um die Softgoal-Erfüllung zu maximieren. Botangen et al. (2018) präsentieren einen Ansatz zur Konfiguration von Zielmodellen nach Präferenzen, die wiederum von dem aktuellen Kontext abhängen (z. B. Standort des Patienten, Wetter etc.). Die Beziehung zwischen Präferenzen und Kontexten wird in einer sogenannten kontextuellen Präferenzspezifikation erfasst, die eine Handlung (in Bezug auf ein Ziel oder eine Aufgabe aus dem Zielmodell), einen Kontext und einen Wert, der die Präferenz für diese Handlung in diesem speziellen Kontext repräsentiert, miteinander verknüpft. Der Kontext, die kontextuelle Präferenzspezifikation und das Zielmodell werden in ein Disjunktive-Logik-Programm übersetzt, das die Werte für jede mögliche Alternative berechnet und so dabei hilft, die optimale Konfiguration zu finden. Ein Ansatz zur Überprüfung, ob Ziele in einem bestimmten Kontext erreichbar sind, wird von Chatzikonstantinou und Kontogiannis (2013) vorgeschlagen. Die Autoren entwickeln ein Rahmenwerk für die Prüfung, ob ein Agent seine Ziele in einem gegebenen Kontext erreichen kann. Zielmodelle werden verwendet, um den internen Entwurf der einzelnen Agenten zu spezifizieren, während Beziehungen anhand von Commitments spezifiziert werden.

DeVries und Cheng (2016) stellen einen Ansatz zur symbolischen Analyse von KAOS-Modellen vor. Jedem Ziel wird zugeordnet, unter welchen Kontextbedingungen es erfüllt werden kann und welcher Sensor diese überwacht. Durch die Identifikation von Fällen, in denen für bestimmte Kontextbedingungen Ziele nicht erfüllt werden, können automatisiert fehlende

Anforderungen erkannt werden. DeVries und Cheng (2017a) erweitern den in DeVries und Cheng (2016) vorgestellten Ansatz zum Einsatz während der Laufzeit, was es erlaubt, auch bei unerwarteten Kontextänderungen fehlende Anforderungen zu identifizieren. In DeVries und Cheng (2017b) wird der Ansatz aus DeVries und Cheng (2016) um Evolutionary Computation erweitert, um repräsentative Beispiele für fehlende Anforderungen zu identifizieren, was eine Korrektur erleichtert.

Der Ansatz von Chen et al. (2014) nutzt Zielmodelle für die Adaptionsplanung selbst adaptiver Systeme. Das Zielmodell definiert alternative Möglichkeiten zur Anforderungserfüllung. Es ist mit einem Designentscheidungsmodell verknüpft, das mögliche Architekturentscheidungen definiert. Wenn das System feststellt, dass eine Adaption erforderlich ist, wird basierend auf den Informationen im Zielmodell und im Designentscheidungsmodell eine angepasste Architektur per Modelltransformation erstellt.

Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext unterstützen zu einem gewissen Grad die Validierung. Dies ist jedoch nicht ihr Hauptzweck. Daher gilt Kriterium V1 als teilweise erfüllt. Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext berücksichtigen Variabilität, jedoch nicht Variabilität, die bezogen auf unterschiedliche SoS-Konfigurationen entsteht. Daher gilt V2 als nicht erfüllt. Im Rahmen der zielorientierten Konfiguration von Systemen können Inkonsistenzen im Zielmodell, im SoS-Modell und zwischen dem Ziel- und SoS-Modell erkannt werden. Jedoch ist nicht sichergestellt, dass dies zuverlässig geschieht. Daher gelten V6, V7 und V8 ebenfalls als teilweise erfüllt. Tabelle 3.23 fasst die Bewertung der Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext zusammen.

Tabelle 3.23: Bewertung – Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	teilweise erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	nicht erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	teilweise erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	teilweise erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	teilweise erfüllt

3.4.5 Ansatz von Metzger et al. zur automatisierten Analyse von Produktlinien- und Softwarevariabilität

Der Ansatz von Metzger et al. (2007) unterstützt die Spezifikation von Produktlinien- und Softwarevariabilität in getrennten, verknüpften Modellen. Die Softwarevariabilität wird in einem Feature-Diagramm und die Produktlinienvariabilität in einem Orthogal Variability

Model spezifiziert. Beide Modelle werden durch Verbindungen miteinander verknüpft, um Zusammenhänge zu spezifizieren. Auf diesen verknüpften Modellen lassen sich automatisierte Analysen durchführen, die es unter anderem erlauben festzustellen, ob die Produktlinie realisierbar ist oder ob überflüssige Features existieren.

Auch wenn der Ansatz von Metzger et al. (2007) nicht für die Validierung von Zielspezifikationen entwickelt wurde, lässt er sich aufgrund der gleichartigen Baumstruktur von Ziel- und Variabilitätsmodellen auf Zielmodelle übertragen. Er kann somit auch einen Beitrag zur Validierung von Zielspezifikationen leisten. Daher ist das Kriterium V1 teilweise erfüllt. Kriterium V2 wird erfüllt, da der Ansatz die unterschiedlichen SoS-Konfigurationen berücksichtigt. Die automatisierten Analysen lassen sich übertragen, um Inkonsistenzen zwischen Ziel- und SoS-Modellen zu erkennen. Daher gilt V8 als erfüllt. Tabelle 3.24 fasst die Bewertung des Ansatzes zur automatisierten Analyse von Produktlinien- und Softwarevariabilität zusammen.

Tabelle 3.24: Bewertung – Ansatz zur automatisierten Analyse von Produktlinien- und Softwarevariabilität

Kriterium		Bewertung
V1	Berücksichtigung von Zielen bei der Validierung	teilweise erfüllt
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	erfüllt
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	nicht erfüllt
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	nicht erfüllt
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	nicht erfüllt
V6	Erkennung von Inkonsistenzen in Zielmodellen	nicht erfüllt
V7	Erkennung von Inkonsistenzen in SoS-Modellen	nicht erfüllt
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	erfüllt

3.5 Zusammenfassung und verfeinerte Problemstellung

Tabelle 3.25 fasst die Bewertung des Standes der Wissenschaft zusammen. Es zeigt sich, dass für viele Kriterien Ansätze existieren, die einen Beitrag zur Zielerreichung dieser Arbeit leisten können.

Es liegen diverse Ansätze für die Spezifikation von Zielen für Systeme mit variablen SoS-Konfigurationen vor. Diese lassen sich verwenden, um Ziele und die zugehörigen SoS-Konfigurationen zu spezifizieren. Ansätze, die eine Relationierung von Ziel- und Variabilitätsmodellen vorschlagen, erscheinen hierbei besonders hilfreich, da hierüber explizit eine Zuordnung von Zielen zu SoS-Konfigurationen erfolgen kann und sich zusätzlich eine Reduktion des Umfangs der zu erstellenden Spezifikation einstellt. Daher basiert die Lösungsidee auf der Spezifikation der Ziele der SoS-Konfigurationen in GRL-Zielmodellen und kardinalitätsbasierten Feature-Modellen.

Zur Validierung von Zielspezifikationen und Spezifikationen von Systemen mit variablen Systemzusammensetzungen existieren Checklisten-basierte Ansätze. Diese können genutzt

werden, um Zielspezifikationen für SoS-Konfigurationen zu validieren. Der in dieser Arbeit vorgeschlagene Lösungsansatz nutzt diese Validierungsansätze jedoch nicht, da er früher ansetzt. Im Rahmen dieser Arbeit wird die Darstellung der zu validierenden Artefakte optimiert, um die Validierung von Zielspezifikationen für SoS-Konfigurationen zu unterstützen. Die existierenden Checklisten-basierten Ansätze können dann genutzt werden, um diese optimierte Darstellung der Artefakte zu prüfen. Diese Prüfung steht jedoch nicht mehr im Fokus dieser Arbeit.

Erfüllbarkeitsanalyseansätze für Zielmodelle und Ansätze zur Analyse von Feature-Modellen können genutzt werden, um Inkonsistenzen in Ziel- bzw. Feature-Modellen zu identifizieren, und somit manche Defekte automatisiert aufdecken. Des Weiteren eignet sich der Ansatz von Metzger et al. (2007) zur Identifikation von Inkonsistenzen zwischen dem Ziel- und SoS-Modell. Daher werden diese Ansätze im Rahmen der Lösungsidee genutzt, um automatisiert Inkonsistenzen und somit Defekte in Zielspezifikationen von SoS-Konfigurationen zu identifizieren. Modelchecking-Ansätze könnten ebenfalls genutzt werden, allerdings würden diese eine zusätzliche Spezifikation des Verhaltens erforderlich machen.

Existierende Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen zielen darauf ab, einzelne SoS-Konfigurationen abzuleiten, die unter vorgegebenen Rahmenbedingungen eine optimale SoS-Konfiguration ermitteln. Dies kann die Prüfung für diese einzelnen SoS-Konfigurationen erleichtern, da diese isoliert betrachtet werden können. Jedoch ist dieses Vorgehen für Systeme mit vielen variablen SoS-Konfigurationen ungeeignet, da nicht jede SoS-Konfiguration einzeln geprüft werden kann. Das gleiche gilt für Ansätze, die Systeme zielorientiert an einen variablen Kontext anpassen, da auch diese Ansätze lediglich die Ableitung einzelner SoS-Konfigurationen unterstützen.

Der Ansatz von Metzger et al. (2007) ermöglicht die Analyse zweier verknüpfter Modelle und kann somit die Validierung von Zielspezifikationen für SoS-Konfigurationen unterstützen, die dieses Prinzip nutzen. Dies ermöglicht die Identifikation von Inkonsistenzen zwischen dem Ziel- und dem SoS-Modell.

Die Analyse des Standes der Wissenschaft hat gezeigt, dass Ansätze existieren, die die Spezifikation von Zielen für variable Systeme wie Systems-of-Systems unterstützen. Insbesondere die Spezifikation von Zielen für Systems-of-Systems mit einem Basis-Zielmodell und einem Variabilitätsmodell zur Spezifikation der SoS-Konfigurationen ermöglichen die Spezifikation der Ziele für unterschiedliche SoS-Konfigurationen auch bei einer großen Anzahl von SoS-Konfigurationen. Fehler wie Inkonsistenzen in der Spezifikation lassen sich durch diverse automatisierte Prüfverfahren aufdecken. Dies ist jedoch nicht auf alle Arten von Fehlern anwendbar. Beispielsweise kann nicht automatisiert geprüft werden, ob ein bestimmtes Ziel für alle SoS-Konfigurationen spezifiziert ist, in denen es gewünscht ist. Solche Entscheidungen können nur manuell getroffen werden. Aufgrund der Komplexität und des Umfangs der Modelle sind die Zusammenhänge jedoch in der Spezifikation nicht leicht zu erkennen. Daher wird Unterstützung benötigt, die die Modelle so aufbereitet, dass die Validierung, d. h. die

Tabelle 3.25: Übersicht Stand der Wissenschaft

Kriterium		Ansätze
S1	Berücksichtigung von Zielen bei der Spezifikation	• Zielmodellierungsansätze für SoS • Ansätze zur Dokumentation von Variabilität in Zielmodellen • Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen • Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen
S2	Berücksichtigung von unterschiedlichen Systemzusammensetzungen	• Ansätze zur Dokumentation von Variabilität in Zielmodellen • Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen • Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen
S3	Berücksichtigung der Relation von Zielen und Systemzusammensetzungen	• Ansätze zur Dokumentation von Variabilität in Zielmodellen • Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen • Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen
S4	Reduktion des Umfangs der zu erstellenden Spezifikation	• Ansätze zur Dokumentation von Variabilität in Zielmodellen • Ansätze zur Verknüpfung von Ziel- und Variabilitätsmodellen • Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen
V1	Berücksichtigung von Zielen bei der Validierung	• Ansätze zur Validierung von Zielmodellen • Erfüllbarkeitsanalyseansätze • Ansätze zum Modelchecking von Zielmodellen • (Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen) • (Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext) • (Ansatz zur automatisierten Analyse von Produktlinien- und Softwarevariabilität)
V2	Berücksichtigung von unterschiedlichen SoS-Konfigurationen bei der Validierung	• Ansätze zur Validierung von Entwicklungsartefakten für variable Systeme • Ansätze zur Analyse von Featuremodellen • (Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen) • Ansatz zur automatisierten Analyse von Produktlinien- und Softwarevariabilität
V3	Unterstützung der Prüfung, ob ein Ziel für genau alle SoS-Konfigurationen spezifiziert wurde	-
V4	Unterstützung der Prüfung, ob für eine SoS-Konfiguration genau alle Ziele spezifiziert sind	-
V5	Vereinfachung der Darstellung der zu validierenden Spezifikation	-
V6	Erkennung von Inkonsistenzen in Zielmodellen	• (Ansätze zur Dokumentation von Variabilität in getrennten Zielmodellen) • Erfüllbarkeitsanalyseansätze • Ansätze zum Modelchecking von Zielmodellen • (Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen) • Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext)
V7	Erkennung von Inkonsistenzen in SoS-Modell	• (Ansätze zur Validierung von Entwicklungsartefakten für variable Systeme) • Ansätze zur Analyse von Featuremodellen • (Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen) • Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext)
V8	Erkennung von Inkonsistenzen zwischen Ziel- und SoS-Modell	• (Ansätze zur zielorientierten Konfiguration von Systemen mit variablen Systemzusammensetzungen) • (Ansätze zur zielorientierten Analyse von Systemen in variablem Kontext) • Ansatz zur automatisierten Analyse von Produktlinien- und Softwarevariabilität

Prüfung gegen die aktuellen Stakeholder-Wünsche, erleichtert wird. Daraus ergibt sich die folgende detaillierte Zielsetzung dieser Arbeit:

Verfeinerte Zielsetzung der Arbeit

Unterstützung der Validierung von Zielspezifikationen für Systems-of-Systems bestehend aus einem Basis-Zielmodell und einem Modell der SoS-Konfigurationen im Hinblick auf die korrekte Spezifikation der Zusammenhänge zwischen Zielen und SoS-Konfigurationen.

Teil II

Lösungsansatz

4 Kapitel
Lösungsidee und Beiträge der Arbeit

Die Lösungsidee basiert auf der Spezifikation von Zielen und SoS-Konfigurationen in getrennten Modellen und der expliziten Verknüpfung dieser Modelle zur Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen. Zur Unterstützung der Validierung derart komplexer Spezifikationen wird die automatisierte Generierung von Sichten vorgeschlagen. Die Sichten heben hervor, welche Ziele für ausgewählte SoS-Konfigurationen spezifiziert wurden und welche SoS-Konfigurationen für ausgewählte Ziele spezifiziert wurden.

Dieses Kapitel stellt die Lösungsidee vor, wobei im Fokus steht, wie diese Lösungsidee auf dem Stand der Wissenschaft aufbaut und welche Beiträge die Arbeit leistet.

4.1 Lösungsidee

Im Rahmen dieser Arbeit wird ein Ansatz zur Unterstützung der Spezifikation und Validierung von Zielen für SoS-Konfigurationen vorgeschlagen. Bei der zielorientierten Entwicklung werden die Ziele des Systems spezifiziert und validiert. Mit der Validierung wird geprüft, ob die spezifizierten Ziele den Wünschen der Stakeholder entsprechen. Für SoS-Konfigurationen gilt, dass nicht alle SoS-Konfigurationen alle Ziele erreichen sollen. Daher ist bei der zielorientierten Entwicklung von Systems-of-Systems zu berücksichtigen, dass die Ziele nicht nur insgesamt, sondern für alle möglichen SoS-Konfigurationen spezifiziert und validiert werden müssen.

Bei der zielorientierten Entwicklung eines System-of-Systems wird bereits in frühen Phasen sichergestellt, dass in allen möglichen SoS-Konfigurationen die Ziele erfüllbar sind, die es in dieser SoS-Konfiguration erfüllen soll.

Um dies zu realisieren, muss a) die Möglichkeit geschaffen werden, dass Ziele zu SoS-Konfigurationen zugehörig spezifiziert werden können, und b), dass die Zuordnung von spezifizierten Zielen zu SoS-Konfigurationen überprüfbar ist.

Abbildung 4.1 illustriert den Lösungsansatz, der auf einem dreischrittigen Verfahren basiert:

1. Spezifikation der Ziele, der SoS-Konfigurationen und der Zusammenhänge zwischen Zielen und SoS-Konfigurationen

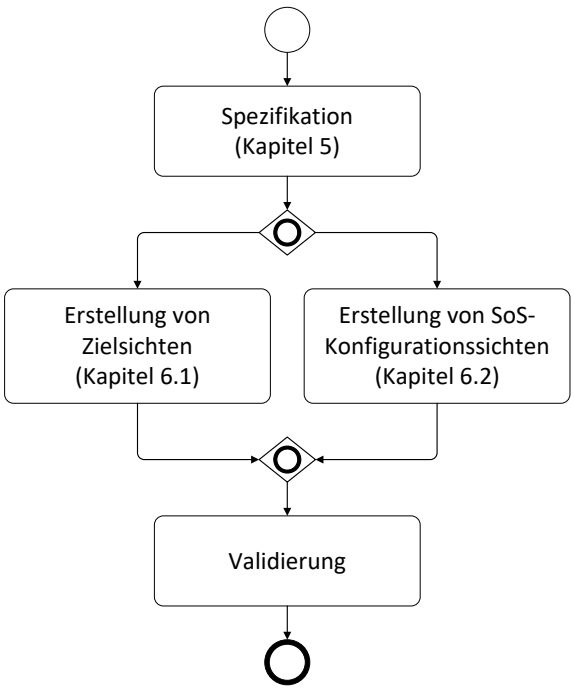


Abbildung 4.1: Überblick über Lösungsidee

Im ersten Schritt werden die Ziele aller SoS-Konfigurationen, die SoS-Konfigurationen selbst sowie die Relationen zwischen Zielen und SoS-Konfigurationen spezifiziert. Dazu werden Ansätze aus der Produktlininenentwicklung (siehe Abschnitt 3.3.2.2) herangezogen. Dieses Vorgehen wird in Kapitel 5 detailliert erläutert.

2. Automatisierte Generierung von Sichten zur Optimierung der Darstellung

   Im zweiten Schritt wird unterschieden, ob der Nutzer eine Zielsicht oder eine SoS-Konfigurationssicht erstellen möchte. Er wählt im Falle einer Zielsicht SoS-Konfigurationen aus. Die Zielsicht wird basierend auf den spezifizierten Informationen automatisiert erstellt. Sie zeigt, welche Ziele für die ausgewählten SoS-Konfigurationen als erfüllbar spezifiziert sind. Die Erstellung von Zielsichten wird in Abschnitt 6.1 vorgestellt. Bei der Erstellung einer SoS-Konfigurationssicht wählt der Nutzer intentionale Elemente im Zielmodell aus. Die SoS-Konfigurationssicht gibt an, für welche SoS-Konfigurationen die ausgewählten Ziele spezifiziert sind. Mit der Erstellung von SoS-Konfigurationssichten befasst sich Abschnitt 6.2. Im Rahmen der Sichtenerstellung werden Konsistenzprüfungen durchgeführt. Dies verhindert die Erstellung ungültiger Sichten, die Widersprüche enthalten.

3. Manuelle Validierung der erzeugten Sichten

   Im dritten Schritt werden die Sichten dahingehend geprüft, ob sie den Stakeholder-Wünschen entsprechen. Mögliche Abweichungen wie das Fehlen von erforderlichen Zielen, das Vorhandensein von nicht erforderlichen Zielen bzw. das Fehlen von erforderlichen SoS-Konfigurationen oder das Vorhandensein von nicht erforderlichen SoS-Konfigurationen deuten auf Defekte in der Spezifikation hin. In diesen Fällen können die in Kapitel 2.4 vorgestellten Ansätze zur Validierung von Anforderungsspezifikationen genutzt werden.

Unter Berücksichtigung des Stands der Wissenschaft schlägt diese Arbeit eine Lösungsidee für die Spezifikation von Zielen von SoS-Konfigurationen und die Unterstützung der Validierung dieser Spezifikation vor, die auf zwei grundlegenden Ideen basiert.

4.1.1 Spezifikation von Zielen und erlaubten SoS-Konfigurationen in getrennten, verknüpften Modellen

Für die Spezifikation von Zielen für SoS-Konfigurationen kommen drei prinzipielle Vorgehensweisen infrage:

1. Für jede SoS-Konfiguration wird ein eigenes Zielmodell erstellt. Dies hat den Nachteil, dass es bei einem System-of-Systems mit vielen SoS-Konfigurationen zu einer komplexen Zielspezifikation kommt. Diese ist schwierig zu überarbeiten, da die gleiche Änderung möglicherweise in viele oder sogar alle Zielmodelle eingearbeitet werden muss. Abbildung 4.2 illustriert dieses Vorgehen.

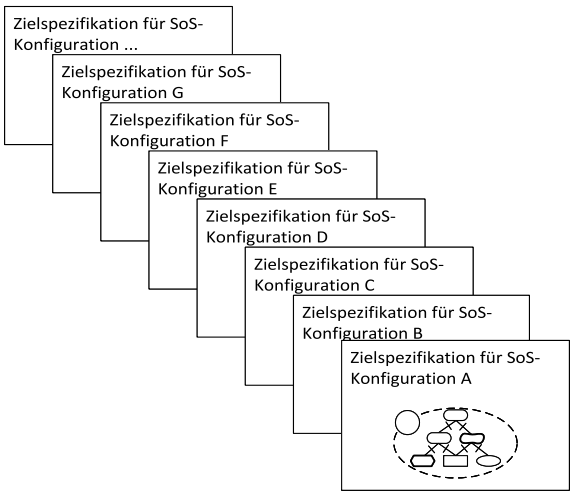


Abbildung 4.2: Eigenes Zielmodell für jede SoS-Konfiguration

2. Es wird ein Zielmodell mit den Zielen für alle SoS-Konfigurationen erstellt und die Gültigkeit für bestimmte SoS-Konfigurationen wird annotiert. Dies hat den Nachteil, dass auch hier bei einem System-of-Systems mit vielen SoS-Konfigurationen ein umfangreiches Zielmodell entsteht. Änderungen an den möglichen SoS-Konfigurationen können zu umfangreichen Änderungen an vielen Annotationen führen. Abbildung 4.3 illustriert dieses Vorgehen.

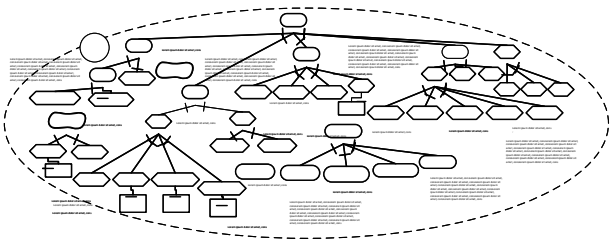


Abbildung 4.3: Ein Zielmodell für alle SoS-Konfigurationen mit Annotationen

3. Es wird ein Zielmodell mit den Zielen für alle SoS-Konfigurationen erstellt sowie ein SoS-Modell, das die möglichen SoS-Konfigurationen spezifiziert, und die Zusammenhänge zwischen Zielen und SoS-Konfigurationen werden durch X-Links spezifiziert. Die Spezifikation von Zielen und SoS-Konfigurationen in getrennten Modellen mit der Spezifikation von X-Links zur Darstellung von Zusammenhängen zwischen Zielen und SoS-Konfigurationen führt hingegen zu einem überschaubaren Umfang der Spezifikation und lässt sowohl Änderungen an den Zielen als auch an den SoS-Konfigurationen und den Zusammenhängen zwischen Zielen und SoS-Konfigurationen mit vertretbarem Aufwand zu. Abbildung 4.4 illustriert dieses Vorgehen.

Zur Unterstützung der Spezifikation von Zielen für ein System-of-Systems unter Berücksichtigung unterschiedlicher SoS-Konfigurationen wird vorgeschlagen, Ziele und SoS-Konfigurationen in getrennten Modellen sowie die Zusammenhänge durch X-Links zu spezifizieren. Hierdurch wird eine übersichtliche Spezifikation von Zielen und SoS-Konfigurationen möglich. Der vorgeschlagene Ansatz baut auf dem Ansatz von Metzger et al. (2007) auf, deren Grundidee aufgegriffen und an die Spezifikation von Zielen, SoS-Konfigurationen und deren Beziehungen angepasst wird. Für die Spezifikation der Ziele wird ein Zielmodell genutzt. Als Modellierungssprache für das Zielmodell wird GRL (International Telecommunication Union 2018) ausgewählt, da sie standardisiert und weitverbreitet ist. Die Spezifikation der SoS-Konfigurationen erfolgt mit einem SoS-Modell, das auf kardinalitätsbasierten Feature-Modellen (Czarnecki et al. 2005) basiert. Das SoS-Modell erlaubt die Berücksichtigung der Anzahl möglicher SoS-Bestandteile in einer SoS-Konfiguration. X-Links zeigen die Zusammenhänge

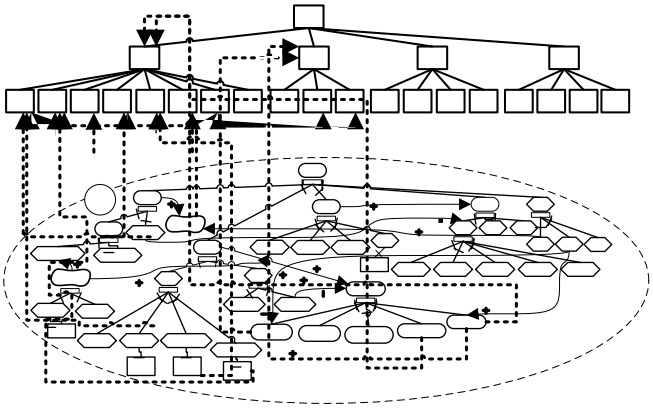


Abbildung 4.4: Verknüpfung von Zielmodell und Modell der SoS-Konfigurationen

zwischen den SoS-Bestandteilen und den Zielen auf. Vorarbeiten hierzu wurden bereits in Brings et al. (2019) veröffentlicht.

4.1.2 Unterstützung der Validierung von Zielspezifikationen für SoS-Konfigurationen durch Sichtenbildung

Da bei der Spezifikation von Zielen für SoS-Konfigurationen Fehler auftreten können, die nicht automatisiert erkennbar sind, ist es notwendig, die Spezifikation zu überprüfen. Dies kann bspw. ein Ziel betreffen, das nicht für eine SoS-Konfiguration spezifiziert wurde, für die es gewünscht ist, oder es wurde ein Ziel für eine SoS-Konfiguration spezifiziert, für die es nicht gewünscht ist.

Diese Überprüfung kann jedoch bei einer Spezifikation von Zielen und SoS-Konfigurationen in getrennten Modellen und der Spezifikation der Zusammenhänge durch X-Links schwierig sein. Im Fall komplexer Zusammenhänge zwischen beiden Modellen ist nicht auf den ersten Blick erkennbar, welche Ziele für welche SoS-Konfigurationen spezifiziert sind. Dadurch erhöht sich die Gefahr, dass Defekte bei der Validierung übersehen werden.

Um die Komplexität zu reduzieren, wird eine automatisierte Sichtengenerierung vorgeschlagen. Zielsichten zeigen, welche intentionalen Elemente für ausgewählte SoS-Konfigurationen spezifiziert sind. SoS-Konfigurationssichten zeigen SoS-Konfigurationen, die für ausgewählte intentionale Elemente aus dem Zielmodell spezifiziert sind. Dies erleichtert die Prüfung, ob Ziele für alle SoS-Konfigurationen spezifiziert sind, für die sie gewünscht sind, und ob für SoS-Konfigurationen alle gewünschten Ziele spezifiziert sind.

Bei der Sichtenbildung können durch Konsistenzprüfungen bereits Fehler automatisiert identifiziert werden, um den Nutzer zu entlasten. Auch bei einer widerspruchsfreien SoS-Zielspezifikation können in den resultierenden Sichten Widersprüche auftreten. Diese Widersprüche deuten auf Defekte in der Zuordnung von Zielen und SoS-Konfigurationen hin. Um

Widersprüche in Zielmodellen zu identifizieren, wird basierend auf dem Ansatz von Amyot et al. (2010) eine Konsistenzprüfung der ausgewählten intentionalen Elemente für die Sichtengenerierung und der generierten Zielsicht vorgeschlagen. Zur Erkennung von Widersprüchen in SoS-Modellen wird basierend auf den Konsistenzprüfungsansätzen von Feature-Modellen (Benavides et al. 2010) eine Konsistenzprüfung der Auswahl der SoS-Konfigurationen für die Sichtengenerierung und der generierten SoS-Konfigurationssicht vorgeschlagen. Diese Konsistenzprüfungen verhindern, dass es bei der Sichtengenerierung zur Erstellung von ungültigen, d. h. inkonsistenten, Sichten kommt. Somit erlauben sie es dem Nutzer, sich bei der Validierung darauf zu konzentrieren, ob die gewünschten Ziele für die SoS-Konfigurationen spezifiziert sind. Vorarbeiten hierzu wurden bereits in Brings und Daun (2019); Brings et al. (2020ab) veröffentlicht.

4.2 Beiträge der Arbeit

In dieser Arbeit wird eine Lösung entwickelt, die die Spezifikation und die Validierung von Zielspezifikationen für ein System-of-Systems mit unterschiedlichen SoS-Konfigurationen unterstützt. Wie in Kapitel 3 verdeutlicht wurde, existieren in der Literatur bereits Ansätze, die einen Beitrag zur Zielsetzung der Arbeit liefern können. Allerdings liegen bisher keine Ansätze vor, die insbesondere die Validierung von Zielspezifikationen für Systeme mit unterschiedlichen SoS-Konfigurationen im Hinblick auf eine Reduktion des Umfangs der zu validierenden Spezifikation und unter Berücksichtigung von Erfüllbarkeitsprüfungen unterstützen. Im Rahmen dieser Arbeit wird ein Lösungsansatz vorgeschlagen, der Sichtengenerierungsansätze und Konsistenzprüfungen so vereint, dass Nutzer gezielt Sichten generieren können. Diese Sichten helfen zu erkennen, ob gewünschte Ziele korrekt spezifiziert sind bzw. ob nicht gewünschte Ziele korrekt nicht spezifiziert sind und ob die spezifizierten Ziele erfüllbar sind. Die Arbeit leistet die folgenden Beiträge:

1. Ein Spezifikationsansatz für Ziele von Systems-of-Systems. Dieser basiert auf dem Ansatz von Metzger et al. (2007), der vorschlägt, Software-Variabilität und Produktlinienvariabilität in getrennten Modellen und die Zusammenhänge zwischen Software-Variabilität durch X-Links zu spezifizieren. In dieser Arbeit wird die Grundidee des Ansatzes aufgegriffen und der Ansatz angepasst, um Ziele und SoS-Konfigurationen in getrennten Modellen zu spezifizieren.

2. Ein Sichtenbildungsansatz für die Erstellung von Sichten auf:

   a) das Zielmodell. Eine Zielsicht zeigt die spezifizierten Ziele für ausgewählte SoS-Konfigurationen.

   b) das SoS-Modell. Eine SoS-Konfigurationssicht zeigt alle möglichen SoS-Konfigurationen für ausgewählten Ziele.

3. Anpassung von Konsistenzprüfungsansätzen für:

   a) Zielmodelle. Hierzu baut die Arbeit auf dem Ansatz von Amyot et al. (2010) auf, der es erlaubt, die Erfüllbarkeit von Zielen basierend auf der Erfüllbarkeit von anderen Zielen zu ermitteln. Der Ansatz wird in dieser Arbeit in den Sichtenbildungsansatz integriert, um a) die Auswahl der intentionalen Elemente des Nutzers auf Konsistenz und b) die generierte Zielsicht auf Konsistenz zu prüfen.

   b) kardinalitätsbasierte Feature-Modelle. Hierzu baut die Arbeit auf dem Ansatz von Weckesser et al. (2016) auf, der es erlaubt, Anomalien in kardinalitätsbasierten Feature-Modellen zu identifizieren. Der Ansatz wird in dieser Arbeit in den Sichtenbildungsansatz integriert, um a) sowohl die Auswahl der SoS-Konfigurationen des Nutzers als auch b) die generierte SoS-Konfigurationssicht auf Konsistenz zu prüfen.

4. Die Evaluation der Anwendbarkeit und der Vorteilhaftigkeit des vorgeschlagenen Ansatzes.

   a) Zur Evaluation der Anwendbarkeit wird der Ansatz an einem Fallbeispiel erprobt und die Sichtengenerierung und Konsistenzprüfung werden prototypisch implementiert.

   b) Zur Evaluation der Vorteilhaftigkeit wird ein kontrolliertes Experiment durchgeführt.

5 Kapitel
Zielspezifikation für Systems-of-Systems

Für die Spezifikation der Ziele für Systems-of-Systems wird eine SoS-Zielspezifikation genutzt. Eine SoS-Zielspezifikation setzt sich aus einem Zielmodell, einem SoS-Modell und X-Links zusammen. Das Zielmodell spezifiziert die Ziele aller SoS-Konfigurationen, das SoS-Modell spezifiziert die Zusammensetzung aller erlaubten SoS-Konfigurationen und die X-Links spezifizieren die Zusammenhänge zwischen Zielen und SoS-Konfigurationen.

Dieses Kapitel befasst sich detailliert mit Schritt 1 des Lösungsansatzes. Es zeigt, wie Ziele und SoS-Konfigurationen spezifiziert werden, um darauf aufbauend die Validierung und Erfüllbarkeitsanalyse zu unterstützten. Die Spezifikation erfolgt mit einer SoS-Zielspezifikation bestehend aus:

- einem Zielmodell
- einem SoS-Modell
- X-Links

Abbildung 5.1 illustriert das Vorgehen bei der Spezifikation. Zuerst werden die Ziele in einem Zielmodell und die zulässigen SoS-Konfigurationen in einem SoS-Modell spezifiziert. Hieran schließt die Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen durch X-Links an.

Zunächst werden die Ziele in einem Zielmodell (siehe Abschnitt 5.1) und die erlaubten SoS-Konfigurationen in einem SoS-Modell spezifiziert (siehe Abschnitt 5.2). Danach werden die Beziehungen zwischen Zielen und SoS-Konfigurationen durch X-Links (siehe Abschnitt 5.3) spezifiziert.

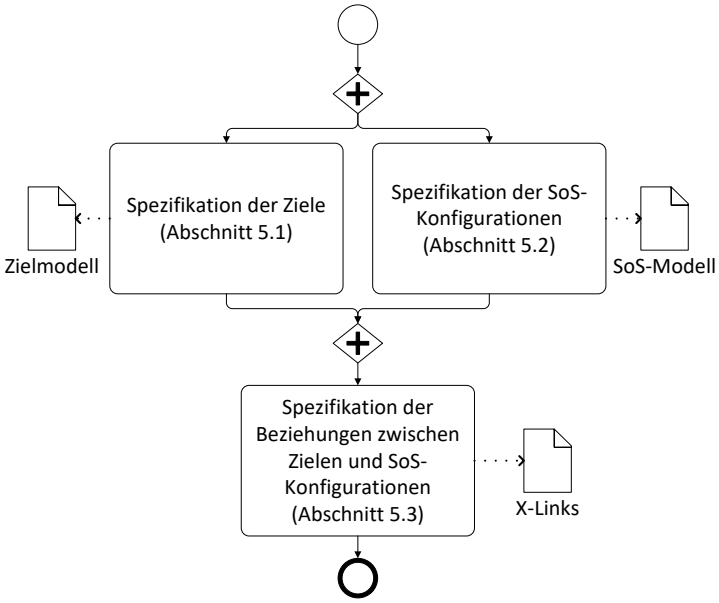


Abbildung 5.1: Spezifikation

5.1 Spezifikation der Ziele in einem Zielmodell

Für die Spezifikation der Ziele wird die in Abschnitt 2.2 vorgestellte Goal-oriented Requirement Language (GRL) (International Telecommunication Union 2018) verwendet. Die Nutzung von GRL erlaubt es, verschiedene Arten von Zielen und Beziehungen zwischen Zielen auszudrücken und die Erfüllbarkeit mithilfe von existierenden Erfüllbarkeitsanalyseansätzen zu prüfen. GRL eignet sich zur Spezifikation der Ziele von Systems-of-Systems. Eine Anpassung ist nicht erforderlich. Spezifiziert werden die Ziele aller erlaubten SoS-Konfigurationen in einem Zielmodell. Für eine Spezifikation der Ziele der Bestandteile des System-of-Systems besteht kein Anlass.

Ein solches Zielmodell zeigt die gewünschten Ziele eines System-of-Systems. Bei der Nutzung von GRL werden die gewünschten Ziele eines System-of-Systems unabhängig von dessen SoS-Konfiguration spezifiziert. Allerdings sieht die GRL keine Berücksichtigung von unterschiedlichen SoS-Konfigurationen vor, weshalb die Nutzung von GRL allein nicht ausreichend ist, um bei der Spezifikation von Zielen für Systems-of-Systems unterschiedliche SoS-Konfigurationen zu berücksichtigen.

5.2 Spezifikation der SoS-Konfigurationen in einem SoS-Modell

Zur Spezifikation der SoS-Konfigurationen wird ein SoS-Modell verwendet. Das SoS-Modell definiert alle erlaubten SoS-Konfigurationen. Es basiert auf den in Abschnitt 2.3 vorgestell-

ten kardinalitätsbasierten Feature-Modellen (Czarnecki et al. 2005). Kardinalitätsbasierte Feature-Modelle eignen sich gut als Grundlage, da sie es erlauben, benötigte und optionale Bestandteile eines System-of-Systems, die mögliche Anzahl von Bestandteilen sowie die Abhängigkeiten zwischen diesen Bestandteilen zu spezifizieren. Dabei repräsentieren die Knoten jedoch keine Features, sondern im Fall von Blättern Systemtypen oder Kategorien von Systemtypen. Systemtypen definieren, welche SoS-Bestandteile Teil des System-of-Systems sein können. Mithilfe von Kategorien können verschiedene Systemtypen mit Ähnlichkeiten zusammengefasst werden. Dadurch lassen sich Einschränkungen spezifizieren, die nicht nur auf einen bestimmten Systemtyp abzielen, sondern auf mehrere ähnliche Systemtypen, die einer Kategorie angehören.

Wie in Abschnitt 2.3 dargestellt wurde, gibt es unterschiedliche Dialekte und konkrete Ausprägungen von Feature-Modellen. In dieser Arbeit, wird – basierend auf der vorgestellten allgemeinen Dialektik – das SoS-Modell definiert, das alle Eigenschaften besitzt, um die Zusammensetzungen von SoS-Konfigurationen zu spezifizieren.

Ein SoS-Modell SM wird wie folgt definiert:

$$SM = \langle N, \kappa, VG, DE, CE \rangle$$

wobei gilt:

- N ist die Menge der Knoten. Für die Menge der Knoten N gilt:

  $$N = \{r\} \cup C \cup ST$$

  Die Wurzel r repräsentiert das System-of-System, bspw. eine Transportroboterflotte. Die Knoten C sind weder Wurzel noch Blätter. Sie stehen für Kategorien von Systemtypen, die Teil des System-of-Systems sein können, bspw. höhenverstellbare Roboter. Die Knoten ST sind Blätter. Sie repräsentieren Systemtypen, deren Instanzen Teil des System-of-Systems sein können, z. B bestimmte Robotertypen.

- κ ist eine Relation, die jedem Knoten $n \in N$ eine Kardinalität $i_F \in I_F$ zuweist.

  $$\kappa : N \to I_F$$

  Die Kardinalität der Wurzel r ist immer [1..1]. Die einer Kategorie zugewiesene Kardinalität gibt an, wie viele zu dieser Kategorie gehörende Systemtypen mindestens und höchstens instantiiert werden dürfen. Dadurch lässt sich bspw. spezifizieren, dass der spezifizierten Transportroboterflotte null bis unendlich viele höhenverstellbare Roboter angehören dürfen. Die einem Systemtyp zugewiesene Kardinalität gibt an, wie oft der Systemtyp mindestens und höchstens instantiiert werden muss, bspw. dass drei bis zehn Roboter vom Typ N Bestandteil der Transportroboterflotte sein dürfen.

- VG ist die Menge von Variabilitätsgruppen. Eine Variabilitätsgruppe vg ermöglicht es, die Anzahl von unterschiedlichen Systemtypen oder Kategorien einer gemeinsamen Oberkategorie zu spezifizieren. Dadurch kann bspw. ausgedrückt werden, dass ein System-of-Systems immer SoS-Bestandteile von zwei unterschiedlichen zur Kategorie B gehörenden Systemtypen enthalten muss. Damit lässt sich bspw. spezifizieren, dass eine Transportroboterflotte aus fünf verschiedenen Arten von Transportrobotern immer zwei bis drei verschiedene Arten von Transportrobotern beinhalten muss. Alle Knoten einer Variabilitätsgruppe müssen den gleichen Elternknoten haben. Eine Variabilitätsgruppe $vg \in VG$ relationiert die Kinderknoten n eines Elternknotens n' zu einer Group-Typ-Kardinalität $i_T \in I_T$. Es gilt:

  $$vg = \langle \{n \in N | n' \rightarrow n\}, i_T \rangle$$

  Hier darf die maximale Kardinalität nicht größer als die Anzahl der Kinderknoten n sein.

- DE ist die Menge der Dekompositionskanten. Die Dekompositionskanten DE verbinden Knoten und zeigen somit die Zugehörigkeit von Systemtypen zu Kategorien auf, z. B. dass es sich bei Robotern vom Typ S um Schwerlastroboter handelt. Es gilt weiterhin:

  $$DE \subseteq N \times N$$

  Die Dekompositionskanten müssen azyklisch sein. Das bedeutet, dass kein Knoten $n \in N$ existiert, der durch eine oder mehrere Dekompositionskanten mit sich selbst verbunden ist. Daher gilt:

  $$\nexists n_1, ..., n_k \in N$$

  $$n_1 \rightarrow ... \rightarrow n_k \rightarrow n_1$$

- CE ist die Menge der Constraintkanten. Die Constraintkanten CE verbinden Knoten, zwischen denen Abhängigkeiten bestehen, bspw. kann ein bestimmter Typ Roboter auch eine bestimmte Art von Ladestation erfordern. Es gibt zwei Arten von Constraints:

  a) Requires-Constraints CE_R

  b) Mutex-Constraints CE_M

  Das heißt:

  $$CE = \{CE_R \cup CE_M\}$$

  Requires-Constraints setzen sich zusammen aus maximal einem Knoten aus N, von dem das Requires-Constraint ausgeht, einer Menge an Knoten aus der Potenzmenge $\mathcal{P}(N)$, auf die sich das Requires-Constraint bezieht, und einer Kardinalität $i_R \in I_R$. Das

bedeutet, dass die Existenz eines Knotens N die Existenz anderer Knoten aus $\mathcal{P}(N)$ in den Grenzen der Kardinalität voraussetzt. Ist kein Ausgangsknoten definiert, handelt es sich um ein globales Constraint, das immer gültig sein muss. In den meisten Fällen geht ein Requires-Constraint von einem Knoten aus und bezieht sich auf einen anderen Knoten. Es gilt:

$$CE_R \subseteq N \times \mathcal{P}(N) \times I_R$$

Im Gegensatz zu den Grundlagen aus Abschnitt 2.3 wird auch die Zuweisung von Kardinalitäten bei Mutex-Constraints unterstützt, um auszudrücken, dass ein Systemtyp oder eine Kategorie eine bestimmte Anzahl eines anderen Systemtyps oder einer anderen Kategorie ausschließt. Dazu werden zwei Kardinalitäten benötigt – eine für jede Richtung, um zu definieren, welche Anzahlkombinationen sich gegenseitig ausschließen.

$$CE_M \subseteq N \times I_R \times N \times I_R$$

Dies erlaubt es bspw. zu spezifizieren, dass drei bis fünf Systeme der Kategorie A und zwei bis sechs Systeme vom Systemtyp B.2 nicht gleichzeitig Teil eines System-of-Systems sein dürfen. Für das Transportroboterbeispiel lässt sich so zum Beispiel spezifizieren, dass niemals drei bis fünf höhenverstellbare Roboter und zwei bis sechs Roboter vom Typ N der Transportroboterflotte angehören sollen.

Abbildung 5.2 zeigt die grafische Darstellung eines SoS-Modells. Das dargestellte SoS-Modell spezifiziert die gültigen SoS-Konfigurationen. Kategorien und Systemtypen sind als rechteckige Kästen dargestellt, Dekompositionskanten als Linien. Es sind fünf verschiedene Systemtypen spezifiziert: $ST_{1.1}, ST_{1.2}, ST_{1.3}, ST_2$ und ST_3. Die Kardinalitäten zeigen, dass jede gültige SoS-Konfiguration mindestens ein System vom Typ ST_2 und ein System vom Typ ST_3 aufweisen muss. Systeme vom Typ $ST_{1.1}, ST_{1.2}$ und $ST_{1.3}$ gehören zur Kategorie C_1. Für die Systeme vom Typ $ST_{1.1}, ST_{1.2}$ und $ST_{1.3}$ ist außerdem eine Variabilitätsgruppe definiert. Eine Variabilitätsgruppe wird durch einen Querstrich, der die Dekompositonskanten der involvierten Knoten verbindet, dargestellt. Es ist zu erkennen, dass bei Systemen vom Typ $ST_{1.1}, ST_{1.2}$ und $ST_{1.3}$ nur Systeme von maximal zwei unterschiedlichen Typen Teil einer SoS-Konfiguration sein dürfen. Constraintkanten verbinden Knoten und geben an, ob es sich um eine Requires- oder eine Excludes-Constraintkante handelt. Die Requires-Kante zeigt hier, dass die Anwesenheit eines Systems vom Typ $ST_{1.2}$ die Anwesenheit mindestens eines Systems vom Typ $ST_{1.3}$ erfordert.

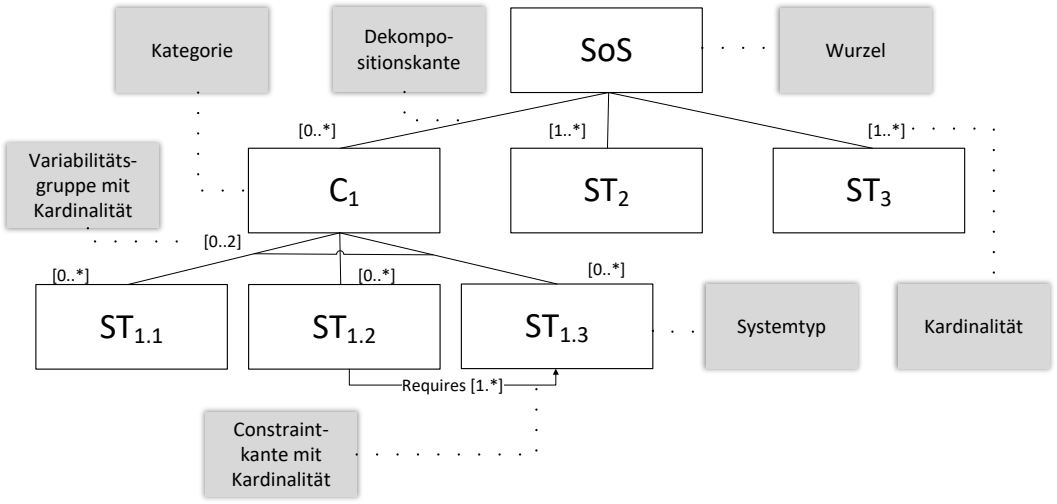


Abbildung 5.2: SoS-Modell

5.3 Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen durch X-Links

Um die Beziehungen zwischen dem Zielmodell und dem SoS-Modell zu dokumentieren, werden X-Links verwendet. Der Einsatz von Feature-Modellen in der Softwareproduktlinienentwicklung erfolgt mit X-Links zur Spezifikation der Beziehungen zwischen Variabilitätsmodellen und Entwicklungsartefakten (Metzger et al. 2007; Bühne et al. 2005). Diese Verbindungen von Variabilitätsmodellen zu – als Basismodell bezeichneten – Entwicklungsartefakten oder Teilen davon relationieren die Entwicklungsartefakte zu den im Variabilitätsmodell spezifizierten Varianten.

Abbildung 5.3 illustriert die Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen durch X-Links. X-Links gehen von einem intentionalen Element im Zielmodell aus und führen zu mindestens einer Kategorie oder einem Systemtyp im SoS-Modell. Es gibt zwei Arten von X-Links. Excludes-X-Links fordern, dass eine bestimmte Anzahl von Systemen nicht Teil des System-of-Systems sein darf. In der Abbildung ist ein Excluces-X-Link ausgehend vom Ziel $G_{1.1}$ zum Systemtyp $ST_{1.1}$ mit einer Kardinalität von [1..*] zu sehen. Eine Kardinalität von [1..*] bedeutet, dass kein System dieses Typs bzw. dieser Kategorie Teil des System-of-Systems sein darf.

Requires X-Links fordern, dass eine bestimmte Anzahl von Systemen eines bestimmten Typs oder einer bestimmten Kategorie Teil des System-of-Systems sein muss. Die Abbildung zeigt einen einfachen Requires-X-Link ausgehend von dem Task $T_{1.1.1}$ zur Kategorie C_1 mit der Kardinalität [2..*]. Die Kardinalität [2..*] bedeutet, dass der Task nur dann erfüllbar ist, wenn das System-of-Systems mindestens zwei Systeme der Kategorie C_1 beinhaltet. Requires-X-Links

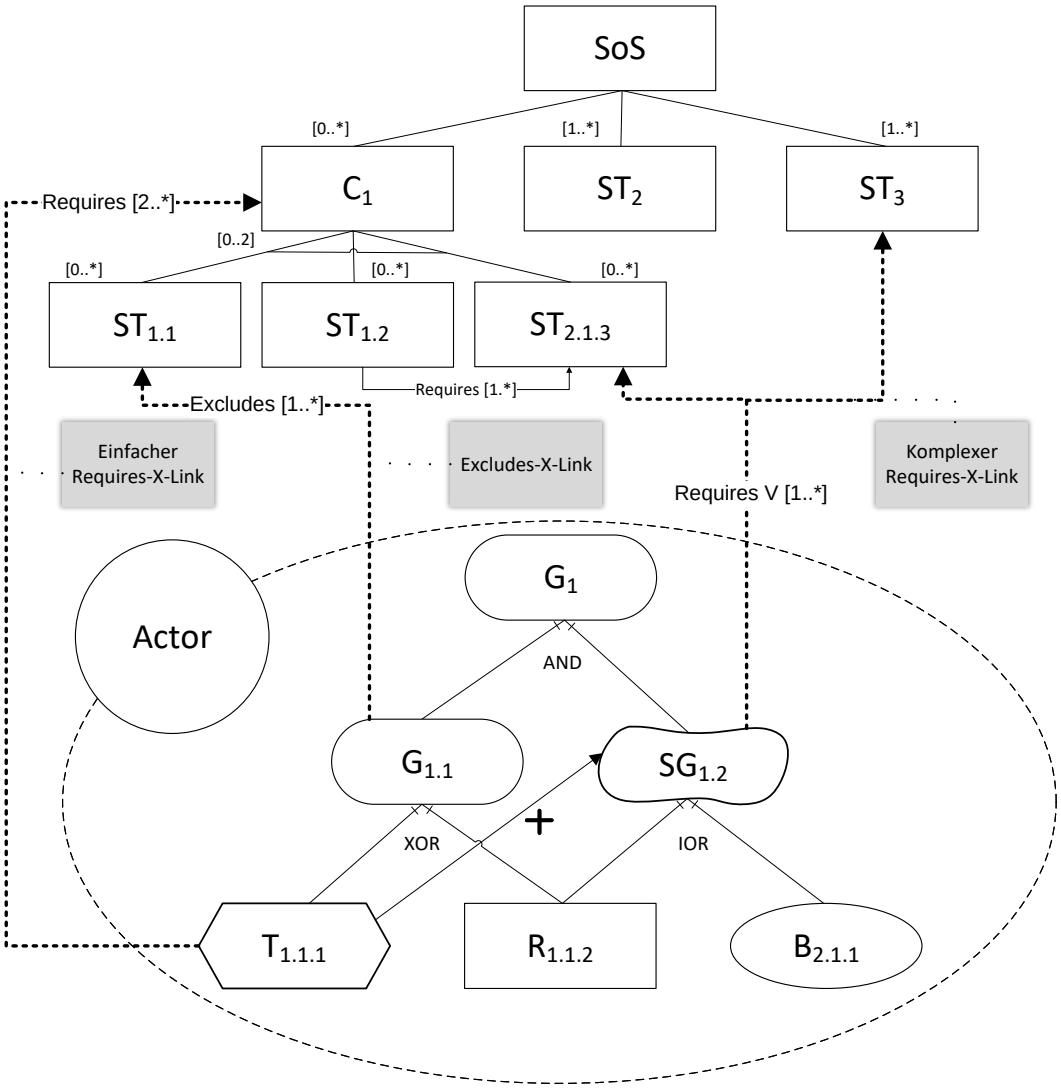


Abbildung 5.3: Spezifikation der Beziehungen zwischen Zielen und SoS-Konfigurationen durch X-Links

lassen sich, wie in der Abbildung dargestellt ist, auch mit mehr als einem Systemtyp oder mehr als einer Kategorie verknüpfen. Dies erlaubt es auszudrücken, dass eine bestimmte Anzahl an Systemen dieser Systemtypen erforderlich ist, auch wenn diese nicht zu einer Kategorie gehören. Aufgrund der erhöhten Komplexität sollten komplexe Requires-X-Links nur sparsam eingesetzt und die Definition der jeweiligen Kategorien möglicherweise vorgezogen werden.

Bei X-Links XL werden mit *requires* oder *excludes* zwei Arten von Beziehungen unterschieden. Es gilt:

$$XL = Req \cup Exc$$

Die Menge der requires-Beziehungen Req definiert Beziehungen, bei denen ein intentionales Element e aus dem Zielmodell GM bestimmte Kategorien C oder Systemtypen ST aus dem SoS-Modell SM erfordert. Zur Vereinfachung der Schreibweise wird die Vereinigung der Menge der Kategorien C und ST mit SC bezeichnet. Es gilt:

$$SC = C \cup ST = N \setminus \{r\}$$

Das heißt,

$$Req = \{\langle e, CS, i_L\rangle | e \in E \wedge CS \subseteq SC \wedge i_L \in I_L\}$$

wobei die X-Link-Kardinalität ein Intervall $i_L \in I_L$ besitzt, bei dem m^{min} die Mindestanzahl der Instanzen des Systemtyps oder der Kategorie und m^{max} die Maximalanzahl der Instanzen des Systemtyps oder der Kategorie bezeichnet. Es gilt:

$$0 \leq m^{min} \leq m^{max}$$

In den meisten Fällen ist der Requires-X-Link nur mit einem Systemtyp oder einer Kategorie verknüpft. Es gilt:

$$req = \langle e, sc, i_L\rangle | e \in E \wedge sc \in SC \wedge i_L \in I_L$$

$e.req$ definiert die Menge der Kategorien und Systemtypen, die ein bestimmtes intentionales Element $e \in E$ erfordert, d. h.

$$e.req = \{sc \mid \exists \langle g, sc, i_L\rangle \in Req\}$$

Dementsprechend kann die Menge der Excludes-Relationen definiert werden als

$$Exc = \{\langle e, sc, i_L\rangle | e \in E \wedge sc \in SC \wedge i_L \in I_L\}$$

$e.exc$ definiert die Menge der Kategorien und Systemtypen, die ein bestimmtes intentionales Element $e \in E$ ausschließt, d. h.

$$e.exc = \{sc \mid \exists \langle g, sc, i_L\rangle \in Exc\}$$

5.4 Beispiel für eine SoS-Zielspezifikation

In diesem Abschnitt wird die Erstellung einer SoS-Zielspezifikation am Beispiel einer Transportroboterflotte veranschaulicht.

Abbildung 5.4 illustriert die Nutzung der GRL für eine Auswahl von Zielen für das Beispiel der Transportroboterflotte. Zunächst ist das Ziel *Warentransport* spezifiziert. Dieses ist in zwei Unterziele dekomponiert: *Gefahren vermeiden* und *Optimierter Warentransport*. Da beide Unterziele erreicht werden müssen, um das Oberziel zu erfüllen, wird eine UND-Dekomposition definiert. *Optimierter Warentransport* ist als Softgoal spezifiziert, da nicht objektiv bestimmt werden kann, wann dieser ausreichend optimiert ist. Das Ziel *Gefahren vermeiden* wird weiter zerlegt, unter anderem in das Ziel *Zusammenstöße vermeiden*. Dieses ist wiederum in die Ziele *Hinderniserkennung* und *Genaue Pfadplanung* gegliedert. Außerdem ist das Softgoal *Geringe Wartungskosten* spezifiziert. Auch hier handelt es sich um ein Softgoal, weil nicht objektiv bestimmt werden kann, wann diese Kosten gering sind. Es ist spezifiziert, dass die Erfüllung des Softgoals *Optimierter Warentransport* einen positiven, aber nicht zur Erfüllung ausreichenden Einfluss auf das Softgoal *Geringe Wartungskosten* ausübt, da der Verschleiß der Roboter durch optimierten Warentransport niedrig gehalten wird. Die Erfüllung der Ziele *Warenübergabe auf variabler Höhe* und *Aufteilung eines Transportauftrags auf mehrere Roboter* nimmt wiederum einen positiven Einfluss auf die Erfüllung des Ziels *Optimaler Warentransport*, da die Erfüllung dieser Ziele eine flexiblere Aufteilung der Transportaufträge unterstützt und somit mehr Möglichkeiten zur Optimierung des Warentransports bietet. Das Ziel *Genaue Pfadplanung* wirkt sich hingegen negativ auf das Softgoal *Autonomie und Flexibilität* aus. Abbildung 5.4 zeigt den beschriebenen Ausschnitt aus dem Zielmodell für eine Transportroboterflotte. Hier ist die Transportroboterflotte, d. h. das System-of-Systems, als ein Akteur dargestellt. Die Ziele, die diesem Akteur zugeordnet sind, sollen von der Transportroboterflotte erfüllbar sein, jedoch nicht unbedingt in jeder SoS-Konfiguration. Daher kann das Zielmodell Ziele enthalten, die nicht in jeder SoS-Konfiguration erreichbar sein müssen. Beispielsweise kann das Ziel *Warenübergabe auf variabler Höhe* lauten.

Abbildung 5.5 zeigt einen beispielhaften Auszug aus einem SoS-Modell für eine mögliche SoS-Konfiguration einer Transportroboterflotte. Es ist spezifiziert, dass die Transportroboterflotte einen bis maximal 20 *Roboter* enthalten muss. Insgesamt gibt es sechs verschiedene Typen von Robotern. Jeder Robotertyp kann prinzipiell gar nicht bis zu 20 Mal Teil der Transportroboterflotte sein. Roboter vom Typ S und vom Typ SF können schwere Lasten transportieren. Roboter vom Typ F und vom Typ FH eignen sich für den Transport von Flüssigkeiten. Roboter H und Roboter FH sind höhenverstellbar. Neben Robotern gehören zur Transportroboterflotte auch *Ladestationen*, *Warenabladestationen* und *Warenaufnahmestationen*. Für jede dieser Kategorien muss ein passendes System Teil der Transportroboterflotte sein. Die Maximalanzahl ist nicht eingeschränkt. Bei den Ladestationen kann zwischen *Allgemeinen Ladestationen* und *Schnellladestationen* unterschieden werden. Es gibt mit D_{40} und D_{60} zwei verschiedene Typen von

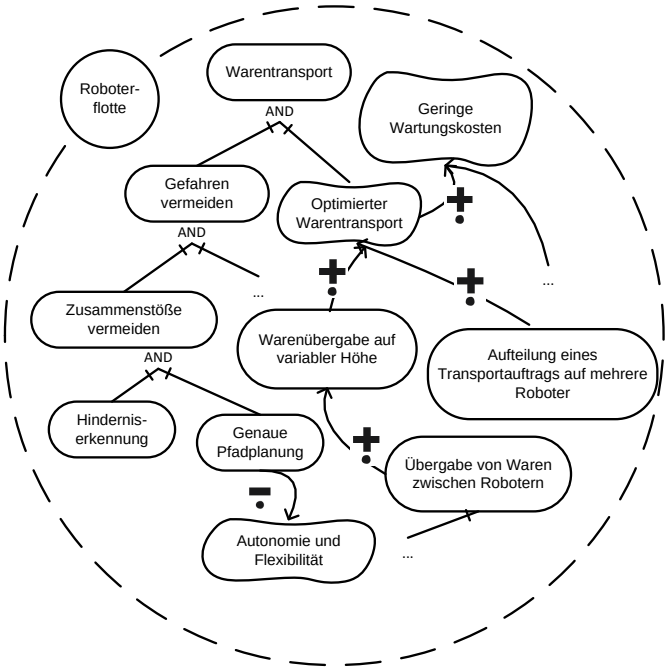


Abbildung 5.4: Ausschnitt aus einem Zielmodell für eine Transportroboterflotte

Warenabladestationen. Bei den *Warenaufnahmestationen* sind ebenfalls zwei verschiedene Typen P_{40} und P_{60} abzugrenzen. Der *Roboter H* erfordert mindestens eine *Schnellladestation* und für Roboter vom Typ S müssen die Warenabladestation D_{40} und die Warenaufnahmestation P_{40} vorhanden sein.

Abbildung 5.6 illustriert die Spezifikation von Zusammenhängen zwischen Zielen und SoS-Konfigurationen an dem Transportroboterflottenbeispiel. Es sind vier Excludes-X-Links, zwei einfache Requires-X-Links und ein komplexer Requires-X-Link spezifiziert. Das Ziel *Zusammenstöße vermeiden* schließt das Vorhandensein von mehr als zehn Robotern in der Flotte aus, da eine hohe Anzahl an Robotern die Gefahr von Zusammenstößen intensiviert. Die Ziele *Aufteilung eines Transportauftrags auf mehrere Roboter* und *Übergabe von Waren zwischen Robotern* erfordern mindestens zwei Roboter in der Transportroboterflotte. Zur Erreichung des Ziels *Warenübergabe auf variabler Höhe* muss entweder mindestens ein Roboter vom Typ H oder vom Typ FH vorhanden sein und das intentionale Element *Geringe Wartungskosten* schließt Roboter vom Typ *SF* und *FH* sowie *Schnellladestationen* aus.

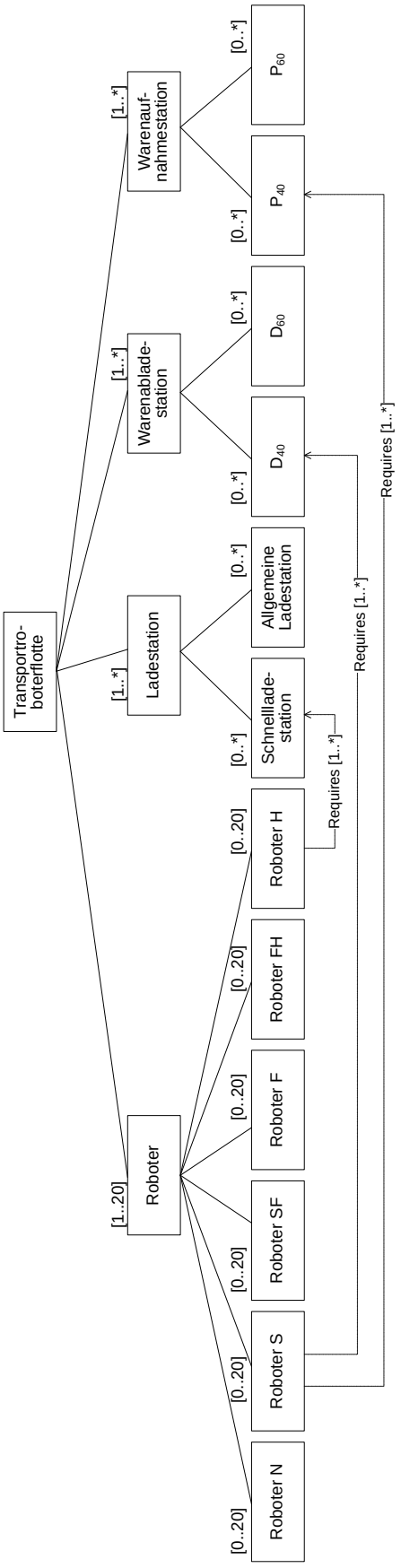


Abbildung 5.5: Ausschnitt aus einem SoS-Modell für eine Transportroboterflotte

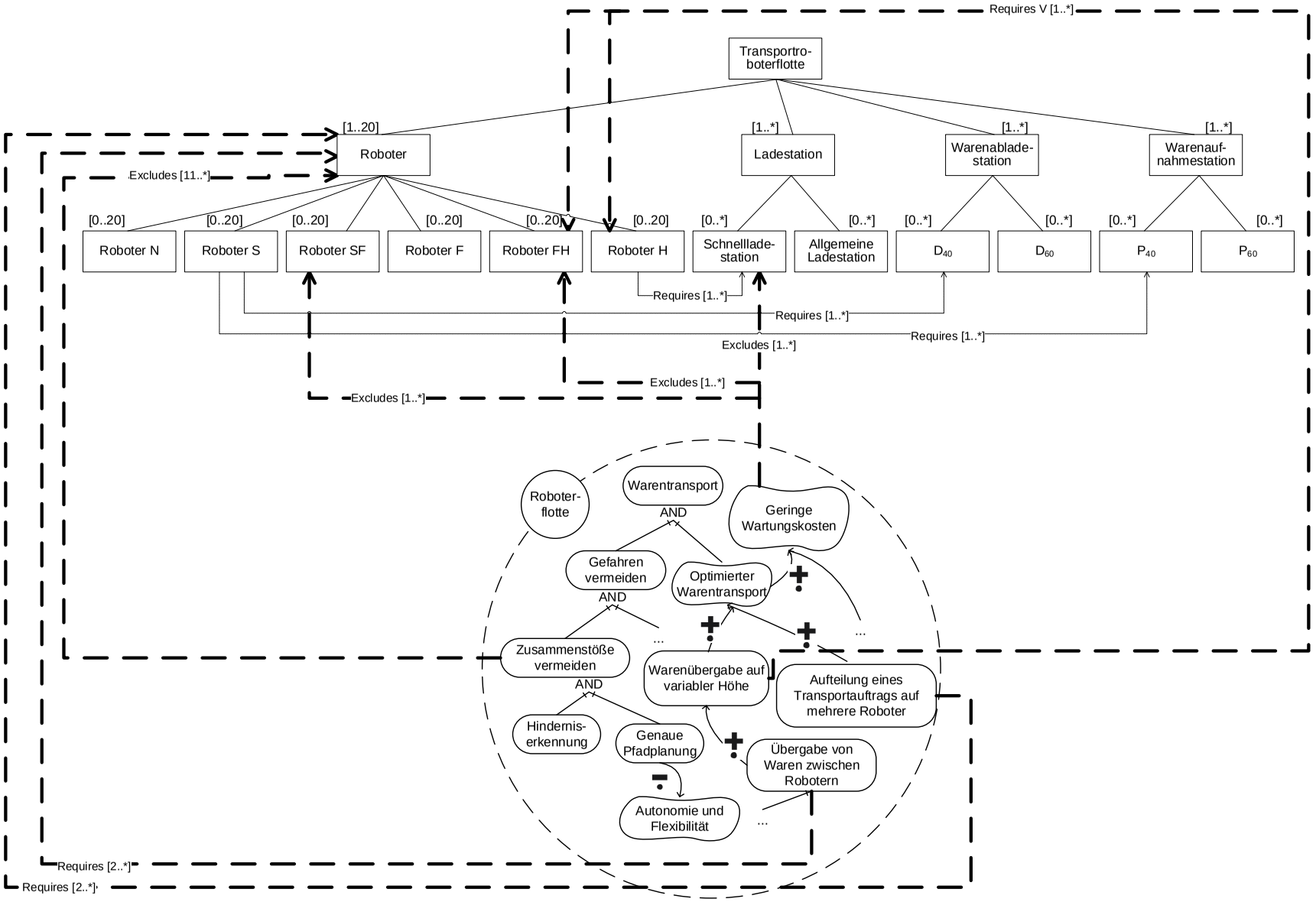


Abbildung 5.6: Relationierung von Zielen und Zusammensetzungen einer Transportroboterflotte

6 Kapitel
Sichtenerstellung

Durch Modelltransformationen werden Sichten erstellt. Zielsichten zeigen, welche Elemente des Zielmodells für ausgewählte SoS-Konfigurationen als erfüllbar spezifiziert sind. Sos-Konfigurationssichten geben wieder, für welche SoS-Konfigurationen ausgewählte Elemente des Zielmodells als erfüllbar spezifiziert sind.

Dieses Kapitel detailliert Schritt 2 des Lösungsansatzes. Es zeigt, wie die Ziel- und SoS-Konfigurationsicht erstellt werden, um die Validierung zu unterstützen. Dabei liegt der Fokus vollständig auf der Erstellung der Ziel- und SoS-Konfigurationssicht. Die im Rahmen der Sichtenerstellung durchgeführten Konsistenzprüfungen werden in Kapitel 7 erläutert.

Der Ansatz unterstützt die Erstellung von Ziel- und SoS-Konfigurationssichten, wobei auch Inkonsistenzen identifiziert werden. Hierdurch wird die Erstellung ungültiger Sichten verhindert. Abschnitt 6.1 stellt zunächst die Erstellung einer Zielsicht vor. Abschnitt 6.2 befasst sich anschließend mit der Erstellung einer SoS-Konfigurationssicht.

Bei der Auswahl der SoS-Konfigurationen zur Erstellung der Zielsicht sowie bei der Erstellung der SoS-Konfigurationssicht können Inkonsistenzen auftreten. Daher ist es erforderlich, sowohl im Rahmen der Erstellung einer Zielsicht als auch einer SoS-Konfigurationssicht SoS-Modelle auf Konsistenz zu prüfen. Diese Prüfung wird in Abschnitt 7.2 beschrieben. Auch bei der Auswahl von intentionalen Elementen im Zielmodell sowie bei der Erstellung der Zielsicht kann es zu Inkonsistenzen kommen. Daher ist es ebenfalls notwendig, sowohl im Rahmen der Erstellung einer Zielsicht als auch einer SoS-Konfigurationssicht die Zielmodelle auf Konsistenz zu prüfen. Diese Prüfung wird in Abschnitt 7.1 beschrieben.

6.1 Erstellung einer Zielsicht

Abbildung 6.1 illustriert die Erstellung einer Zielsicht. Im ersten Schritt wählt der Nutzer im SoS-Modell aus, für welche SoS-Konfigurationen eine Zielsicht erstellt werden soll. Diese Auswahl kann eine oder mehrere SoS-Konfigurationen beinhalten. Basierend auf den ausgewählten SoS-Konfigurationen im SoS-Modell, dem Zielmodell und den X-Links wird im zweiten Schritt eine Zielsicht generiert. Die Zielsicht zeigt, welche intentionalen Elemente für die ausgewählten SoS-Konfigurationen als erfüllbar und welche als nicht erfüllbar spezifiziert sind.

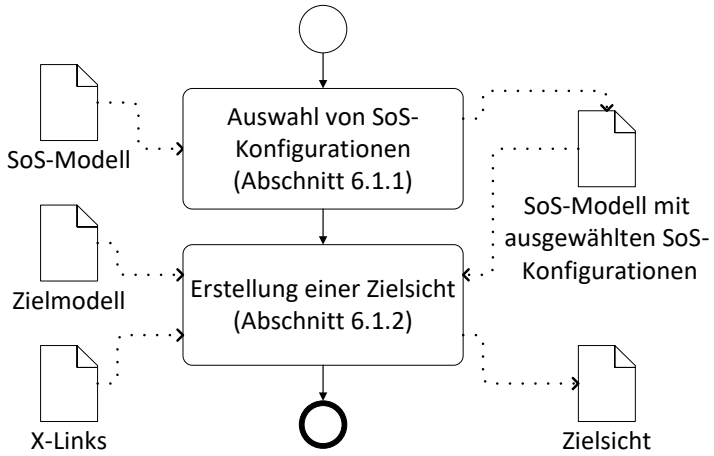


Abbildung 6.1: Erstellung einer Zielsicht

Bei einer Zielsicht GM' handelt es sich um eine Projektion auf das Zielmodell GM des System-of-Sytems.

$$\rho : (GM, SM', XL) \rightarrow GM'$$

Dabei wird die Sicht GM' unter Berücksichtigung eines durch den Anwender selektierten SoS-Modells SM' gebildet. SM' unterscheidet sich vom ursprünglichen SoS-Modell SM durch die vom Nutzer ausgewählten Kardinalitäten κ':

$$SM' = \langle N, \kappa', VG, DE, CE \rangle$$

Die Sicht selbst kann ebenfalls als ein GRL-Zielmodell beschrieben werden. Daher ist eine Zielsicht GM' analog zu der in Abschnitt 2.2 formulierten Definition eines Zielmodells definiert als

$$GM' = \langle A, E', L, \alpha \rangle$$

Bei der Zielsicht GM' handelt es sich um eine grafische Hervorhebung verschiedener intentionaler Elemente – abhängig von der Auswahl im SoS-Modell. Da keine neuen Elemente erzeugt werden, unterscheidet sich GM' von GM nur in der Menge E'. Bei der Menge E' werden sämtliche Elemente $e \in E$ abhängig von ihrer Erfüllbarkeit unterschiedlichen Klassen zugeordnet.

Die Menge E'_S enthält die intentionalen Elemente, die für alle ausgewählten SoS-Konfigurationen erfüllbar sind. Der Menge E'_N gehören die intentionalen Elemente an, die für alle ausgewählten SoS-Konfigurationen nicht erfüllbar sind. Die Menge E'_P umfasst die intentionalen Elemente, die für eine Teilmenge der ausgewählten SoS-Konfigurationen erfüllbar sind. Die Menge E'_D enthält alle intentionalen Elemente, deren Erfüllbarkeit nicht von den

ausgewählten SoS-Konfigurationen abhängig ist. Es gilt:

$$E' = E'_S \cup E'_N \cup E'_P \cup E'_D$$

6.1.1 Auswahl der SoS-Konfigurationen

Um eine Sicht auf das Zielmodell zu erstellen, wählt der Nutzer die SoS-Konfigurationen aus, für die die erfüllbaren intentionalen Elemente angezeigt werden sollen. Dazu definiert er ein selektiertes SoS-Modell SM', das aus dem ursprünglichen SoS-Modell SM abgeleitet wird. Durch eine Anpassung der Kardinalitäten drückt der Nutzer aus, welche Anzahl von Systemen von Kategorien oder Systemtypen bei der Sichtenbildung berücksichtigt werden muss. Hierzu definiert der Nutzer neue Kardinalitäten I' für das selektierte SoS-Modell SM'.

Abbildung 6.2 zeigt beispielhaft die Auswahl von SoS-Konfigurationen. Das spezifizierte System-of-Systems kann fünf verschiedene Arten von Systemen ($ST_{1.1}, ST_{1.2}, ST_2, ST_{3.1}$, $ST_{3.2}$) enthalten. Für jeden Systemtyp wurde durch eine Kardinalität spezifiziert, wie viele Systeme dieses Typs ein System-of-Systems mindestens enthalten muss und maximal enthalten darf. Die Kardinalität von Systemtyp ST_2 beträgt $[1..*]$ und legt somit fest, dass jedes dieser System-of-Systems mindestens ein System vom Typ ST_2 enthalten muss und dass dieses beliebig oft enthalten sein kann. Die Kardinalitäten der anderen Systemtypen betragen $[0..*]$. Daher können Systeme dieser Typen ebenfalls beliebig oft enthalten sein. Des Weiteren ist spezifiziert, dass Systeme vom Typ $ST_{1.1}$ und $ST_{1.2}$ zur Kategorie C_1 und die Systeme vom Typ $ST_{3.1}$ und $ST_{3.2}$ zur Kategorie C_3 gehören. Für diese beiden Kategorien ist wieder jeweils spezifiziert, wie viele Systeme dieser Kategorie mindestens und maximal in dem System-of-Systems vorhanden sein müssen bzw. dürfen. Von Systemen der Kategorie C_1 muss das System-of Systems keine enthalten und kann es beliebig viele enthalten. Von Systemen der Kategorie C_3 muss das System-of-Systems mindestens eins enthalten. Außerdem kann es beliebig viele Systeme der Kategorie C_3 umfassen. Zur Auswahl von SoS-Konfigurationen für die Generierung der Zielsicht ändert der Nutzer die Kardinalitäten des ursprünglichen SoS-Modell ab. Damit legt er fest, für welche SoS-Konfigurationen er eine Sicht generieren möchte. In der Abbildung ist dies durch die durchgestrichenen ursprünglichen Kardinalitäten und hinzugefügte Selektionskardinalitäten I' dargestellt. Es ist spezifiziert, dass eine Sicht für SoS-Konfigurationen erstellt werden soll, die ein bis drei Systeme vom Typ $ST_{1.1}$ und zwei bis vier Systeme vom Typ $ST_{3.3}$ enthält.

Daher gilt:

$$i'_{C_1} = [1..3]$$

$$i'_{ST_{3.2}} = [2..4]$$

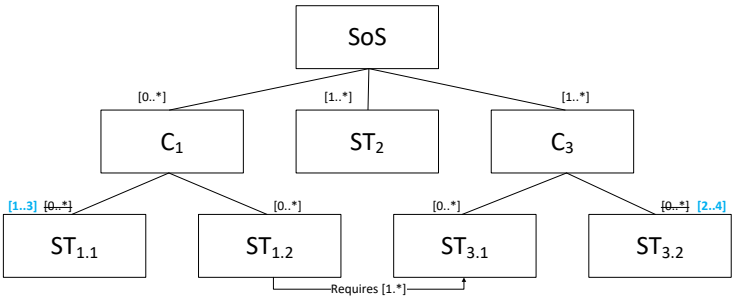


Abbildung 6.2: Auswahl der SoS-Konfigurationen

6.1.2 Generierung von Sichten auf das Zielmodell

Durch die Generierung von Sichten auf das Zielmodell lässt sich untersuchen, welche intentionalen Elemente eine bestimmte SoS-Konfiguration oder eine Menge bestimmter SoS-Konfigurationen erfüllen kann. Dies erlaubt es zu prüfen, ob für eine bestimmte Menge von SoS-Konfigurationen die intentionalen Elemente erfüllbar sind, die sie erfüllen soll. Daher wird auf Basis einer Menge ausgewählter SoS-Konfigurationen eine Sicht auf das Zielmodell abgeleitet.

Sichten auf das Zielmodell werden durch eine Projektion auf das ursprünglich spezifizierte Zielmodell erstellt. Algorithmus 1 definiert die Erzeugung einer Zielsicht, die hervorhebt, welche intentionalen Elemente für alle, manche und keine der ausgewählten SoS-Konfigurationen als erfüllbar spezifiziert sind.

Als Eingabe benötigt der Algorithmus das ursprüngliche Zielmodell GM, die X-Links XL zwischen Elementen des Zielmodells und des SoS-Modells sowie die ausgewählten SoS-Konfigurationen SM'. Als Ausgabe erzeugt der Algorithmus eine Sicht auf das Zielmodell GM'.

Die Einordnung der intentionalen Elemente als erfüllbare, teilweise erfüllbare und nicht erfüllbare intentionale Elemente basiert auf dem Vergleich der Kardinalitäten der X-Links und der ausgewählten Kardinalitäten des selektierten SoS-Modells. Daher werden bei der Erstellung einer Zielsicht die X-Links durchlaufen.

Eine Iteration über alle Exclude-X-Links $exc \in EXC$ vergleicht die Kardinalitäten des X-Links i_{exc} mit denen der Kategorie oder des Systemtyps $i_{exc.sc}$. Dadurch lässt sich ermitteln, ob für die gewählten SoS-Konfigurationen das mit dem Exclude-X-Link verbundene intentionale Element $exc.e$ spezifiziert ist. Wenn sich die Kardinalitäten der Kategorie oder des Systemtyps $i_{exc.sc}$ aus einer Submenge der Kardinalitäten des Exclude-X-Links i_{exc} bilden, ist das intentionale Element $exc.e$ für alle ausgewählten SoS-Konfigurationen nicht erfüllbar und wird somit der Menge der nicht erfüllbaren intentionalen Elemente E'_N zugeordnet. Falls es sich bei dem intentionalen Element $exc.e$ nicht um ein nicht erfüllbares intentionales Element handelt, wird geprüft, ob es teilweise erfüllbar ist. Dies trifft zu, wenn die Kardinalitäten der

Kategorie oder des Systemtyps $i_{exc.sc}$ geschnitten mit den Kardinalitäten des Exclude-X-Links i_{exc} keine leere Menge ergibt. Das bedeutet, das intentionale Element $exc.e$ ist für manche der ausgewählten SoS-Konfigurationen erfüllbar. Folglich wird das intentionale Element $exc.e$ der Menge der teilweise erfüllbaren intentionalen Elementen E'_P zugeordnet. Demzufolge wurden durch die Anpassung der Kardinalitäten sowohl SoS-Konfigurationen ausgewählt, für die das intentionale Element spezifiziert ist, als auch SoS-Konfigurationen, für die das intentionale Element nicht spezifiziert ist. Sollte das intentionale Element $exc.e$ weder nicht erfüllbar noch teilweise erfüllbar sein, ist es für alle ausgewählten SoS-Konfigurationen erfüllbar und wird somit der Menge der erfüllbaren intentionalen Elemente E'_S zugeordnet.

Bei einer Iteration über alle Requires-X-Links $req \in REQ$ müssen die Minimal- und Maximalkardinalitäten aufsummiert werden, um zu prüfen, ob die Kardinalitäten im selektierten SoS-Modell so gewählt wurden, dass die Bedingung des Requires-X-Link erfüllt ist. Hierfür werden die Hilfsvariablen $Lower$ und $Upper$ definiert. In einer Schleife summiert $Lower$ die Minimalkardinalität aller Systemtypen oder Kategorien auf, auf die der Requires-X-Link verweist. $Upper$ summiert die Maximalkardinalitäten aller Systemtypen oder Kategorien auf, auf die der Requires-X-Link verweist. Diese werden benötigt, um zu ermitteln, inwiefern die durch den Requires-X-Link geforderten Kardinalitäten mit den ausgewählten Kardinalitäten übereinstimmen.

Danach wird die Kardinalität des Requires-X-Links $i.req$ mit den Werten der Variablen $Lower$ und $Upper$ verglichen. Damit das intentionale Element $req.e$ für alle ausgewählten SoS-Konfigurationen erfüllbar ist, müssen im selektierten SoS-Modell mindestens gleich viele Systemtypen oder Kategorien ausgewählt sein wie die vom Requires-X-Link geforderte Minimalkardinalität $min(i.req)$ und maximal so viele wie die vom Requires-X-Link geforderte Maximalkardinalität $max(i.req)$. Wenn die Minimalkardinalität des Requires-X-Links $min(i.req)$ gleich oder größer der Summe der Minimalkardinalität aller Systemtypen oder Kategorien ist, auf die der Requires-X-Link req verweist, und die Maximalkardinalität des Requires-X-Links $max(i.req)$ gleich oder größer der Summe der Maximalkardinalitäten aller Systemtypen oder Kategorien ist, auf die der Requires-X-Link req verweist, ist das intentionale Element $req.e$ für alle ausgewählten SoS-Konfigurationen erfüllbar. Folglich wird es der Menge der erfüllbaren intentionalen Elemente E'_S zugeordnet.

Das intentionale Element $req.e$ ist nur für einen Teil der ausgewählten SoS-Konfigurationen erfüllbar, wenn entweder die Minimalkardinalität des Requires-X-Links $min(i.req)$ gleich oder größer der Summe der Minimalkardinalitäten aller Systemtypen oder Kategorien ist, auf die der Requires-X-Link req verweist, oder wenn die Maximalkardinalität des Requires-X-Links $max(i.req)$ gleich oder größer der Summe der Maximalkardinalitäten aller Systemtypen oder Kategorien ist, auf die der Requires-X-Link req verweist. In diesem Fall ist das intentionale Element $req.e$ für manche der ausgewählten SoS-Konfigurationen erfüllbar und für manche nicht. Es wird somit der Menge der teilweise erfüllbaren intentionalen Elemente E'_P zugeordnet.

Anderenfalls ist das intentionale Element $req.e$ für alle ausgewählten SoS-Konfigurationen nicht erfüllbar und wird somit der Menge der nicht erfüllbaren intentionalen Elemente E'_N zugeordnet.

Um sicherzustellen, dass jedes intentionale Element nur einer Untermenge zugeordnet ist, wird zunächst für alle erfüllbaren intentionalen Elemente $e' \in E'_S$ überprüft, ob sie auch der Menge der teilweise erfüllbaren intentionalen Elemente E'_P, bzw. der Menge der nicht erfüllbaren intentionalen Elemente E'_N zugeordnet wurden. In beiden Fällen wird das jeweilige intentionale Element aus der Menge der erfüllbaren intentionalen Elemente E'_S entfernt. Desgleichen wird für jedes teilweise erfüllbare intentionale Element $e \in E'_P$ überprüft, ob es auch der Menge der nicht erfüllbaren intentionalen Elemente E'_N zugeordnet wurde. In diesen Fällen wird das jeweilige intentionale Element aus der Menge der teilweise erfüllbaren intentionalen Elemente E'_P entfernt. Dies verhindert, dass intentionale Elemente mit mehreren X-Links unterschiedlichen Mengen zugeordnet werden. Gelangt die Auswertung aller X-Links eines intentionalen Elements zu unterschiedlichen Ergebnissen, wobei es unerheblich ist, ob das intentionale Element erfüllt ist oder nicht, zählt jeweils der schwächste Erfüllungsgrad. D. h. bspw., dass ein intentionales Element, das gemäß der Auswertung von zwei seiner X-Links als erfüllt gilt, während aber ein dritter X-Link zu einer nicht erfüllten Auswertung führt, insgesamt als nicht erfüllt gilt.

Alle intentionalen Elemente, die über keinen X-Link verfügen und daher nicht in die Mengen der erfüllbaren E'_S, teilweise erfüllbaren E'_P oder nicht erfüllbaren E'_N der intentionalen Elemente eingeordnet wurden, werden abschließend der Menge an intentionalen Elementen E'_D zugeordnet, deren Erfüllbarkeit nicht von den SoS-Konfigurationen abhängig ist.

Algorithmus 1 Erstellung Zielsicht

1: **for all** $exc \in EXC$ **do**
2: **if** $i_{exc.sc} \subseteq i_{exc}$ **then**
3: $E'_N = E'_N \cup \{exc.e\}$
4: **else if** $i_{exc.sc} \cap i_{exc} \neq \emptyset$ **then**
5: $E'_P = E'_P \cup \{exc.e\}$
6: **else**
7: $E'_S = E'_S \cup \{exc.e\}$
8: **for all** $req \in REQ$ **do**
9: $Lower = 0$
10: $Upper = 0$
11: **for all** $sc \in req.sc$ **do**
12: $Lower = Lower + min(i_{sc})$
13: $Upper = Upper + max(i_{sc})$
14: **if** $Lower \geq min(i.req) \wedge Upper \leq max(i.req)$ **then**
15: $E'_S = E'_S \cup \{req.e\}$

16: **else if** $Lower \geq min(i.req) \vee Upper \leq max(i.req)$ **then**
17: $E'_P = E'_P \cup \{req.e\}$
18: **else**
19: $E'_N = E'_N \cup \{req.e\}$
20: $E'_S = E'_S \setminus (E'_P \cup E'_N)$
21: $E'_P = E'_P \setminus E'_N$
22: $E'_D = E \setminus (E'_S \cup E'_P \cup E'_N)$

6.1.3 Beispielhafte Erstellung einer Zielsicht

Die Abbildungen 6.3 und 6.4 illustrieren die Generierung einer Zielsicht an dem in Kapitel 5 vorgestellten Beispiel der Transportroboterflotte. Abbildung 6.3 zeigt das SoS-Modell für die ausgewählten SoS-Konfigurationen. Bei der Auswahl wurden im Vergleich zu der ursprünglichen Spezifikation der SoS-Konfigurationen zwei Änderungen vorgenommen. Diese sind blau markiert. Der Transportroboterflotte muss mindestens ein Roboter vom Typ SF angehören. Außerdem muss die Transportroboterflotte genau eine allgemeine Ladestation enthalten. Abbildung 6.4 zeigt die generierte Zielsicht für die ausgewählten SoS-Konfigurationen aus Abbildung 6.3. Nicht erfüllbare intentionale Elemente sind rot dargestellt. Intentionale Elemente, die nur für einen Teil der ausgewählten SoS-Konfigurationen erfüllbar sind, sind gelb unterlegt. Bei diesem Beispiel kann die Transportroboterflotte mit keiner der ausgewählten SoS-Konfigurationen das Softgoal *Geringe Wartungskosten* erreichen. Die intentionalen Elemente *Warenübergabe auf variabler Höhe, Aufteilung eines Transportroboters auf mehrere Roboter, Übergabe von Waren zwischen Robotern* und *Zusammenstöße vermeiden* sind nicht für alle ausgewählten SoS-Konfigurationen erreichbar, sondern nur für einen Teil.

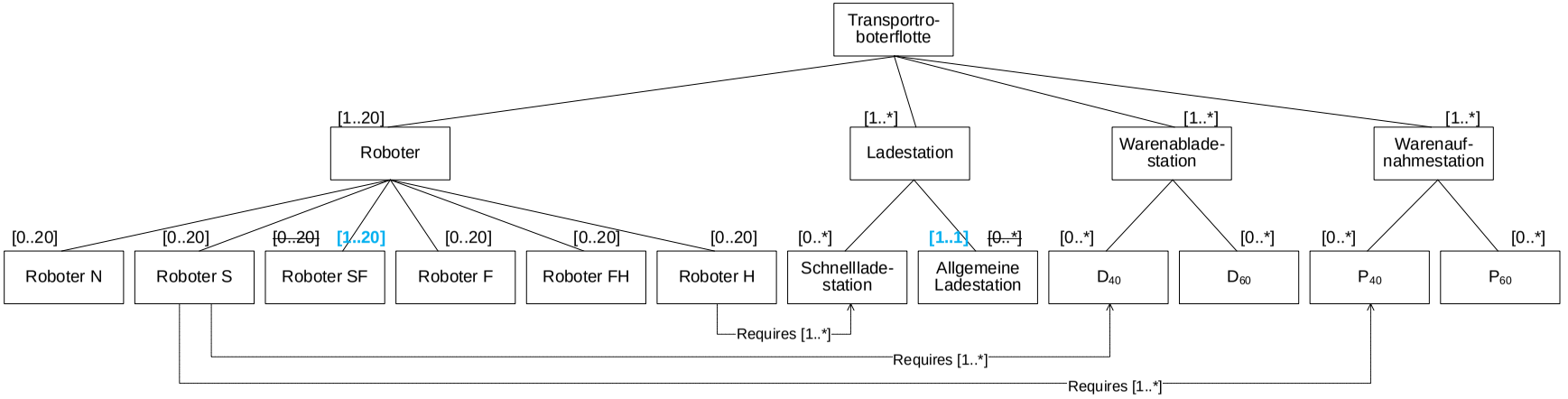


Abbildung 6.3: Beispiel für ein selektiertes SoS-Modell zur Auswahl von SoS-Konfigurationen

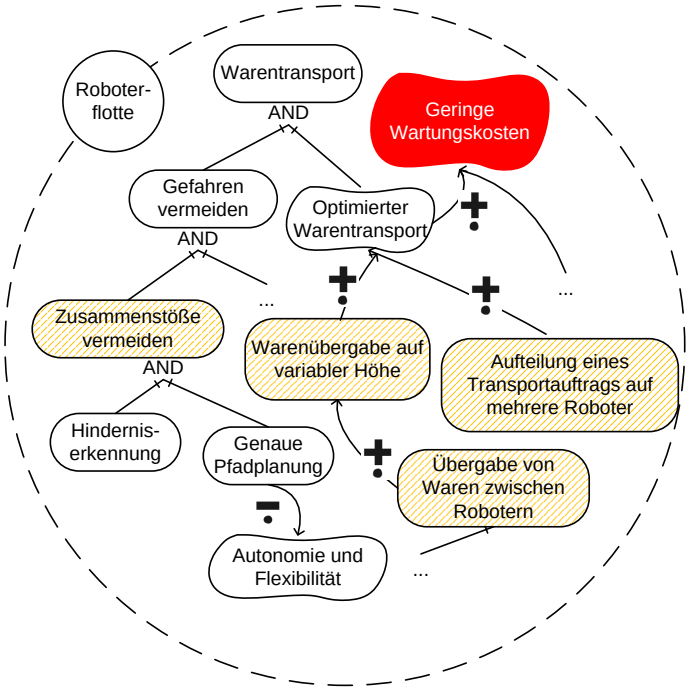


Abbildung 6.4: Beispiel für eine generierte Zielsicht basierend auf den ausgewählten SoS-Konfigurationen

6.2 Erstellung einer SoS-Konfigurationssicht

Abbildung 6.5 illustriert die Erstellung einer SoS-Konfigurationssicht. Im ersten Schritt wählt der Nutzer im Zielmodell aus, welche intentionalen Elemente für die zu erstellende Sicht erfüllbar sein sollen und welche nicht. Basierend auf den ausgewählten intentionalen Elementen im Zielmodell, dem SoS-Modell und den X-Links wird im zweiten Schritt eine SoS-Konfigurationssicht generiert. Die SoS-Konfigurationssicht zeigt ein SoS-Modell, dem entnommen werden kann, für welche SoS-Konfigurationen die ausgewählten intentionalen Elemente spezifiziert sind.

Bei einer SoS-Konfigurationssicht SM' handelt es sich um eine Projektion auf das SoS-Modell SM des System-of-Sytems.

$$\rho : (GM', SM, XL) \rightarrow SM'$$

Dabei wird die Sicht SM' unter Berücksichtigung der durch den Anwender ausgewählten intentionalen Elemente aus dem Zielmodell GM' gebildet. GM' unterscheidet sich vom ursprünglichen SoS-Modell GM durch die vom Nutzer ausgewählten intentionalen Elemente E':

$$GM' = \langle A, E', L, \alpha \rangle$$

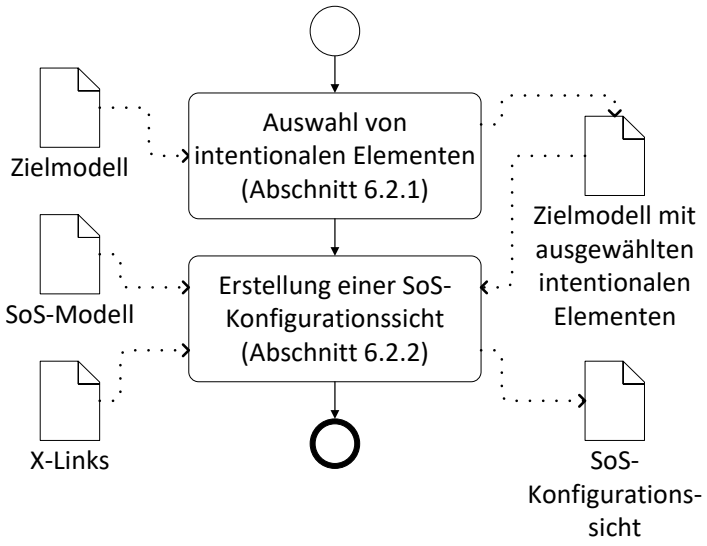


Abbildung 6.5: Erstellung einer SoS-Konfigurationssicht

Die Sicht selbst kann ebenfalls als ein SoS-Modell beschrieben werden. Daher ist eine SoS-Konfigurationssicht SM' analog zu der in Abschnitt 5.2 formulierten Definition eines SoS-Modells definiert als

$$SM' = \langle N, \kappa', VG, DE, CE' \rangle$$

Bei der SoS-Konfigurationssicht SM' handelt es sich um eine Anpassung der Kardinalitäten für Kategorien und Systemtypen – abhängig von der Auswahl der intentionalen Elemente im Zielmodell. Außerdem können sich bei der Sicht Constraints ergeben, weshalb die Menge der Constraints CE' möglicherweise von der ursprünglichen Menge der Constraints CE abweicht.

6.2.1 Auswahl der intentionalen Elemente

Um eine Sicht auf das SoS-Modell zu erstellen, wählt der Nutzer die intentionalen Elemente im Zielmodell aus, die erfüllbar bzw. nicht erfüllbar sein sollen. Dazu teilt er die Menge der intentionalen Elemente in drei Submengen ein: gewünschte Elemente E^W, nicht gewünschte Elemente E^N und Elemente E^D, die weder gewünscht noch nicht gewünscht sind. Es gilt:

$$E' = E^W \cup E^N \cup E^D$$

Jedes intentionale Element muss genau einer dieser drei Mengen zugeordnet werden.

$$E^W \cap E^N = E^W \cap E^D = E^N \cap E^D = \emptyset$$

Abbildung 6.6 zeigt beispielhaft die Auswahl von intentionalen Elementen. Gewünschte Elemente sind mit ✓und nicht gewünschte Elemente mit ✗markiert. $G_{1.1}$ und $R_{1.1.2}$ sind

gewünscht, $B_{2.1.2}$ ist nicht gewünscht. Daher gilt:

$$E^W = \{SG_{1.2}, R_{1.1.2}\}$$

$$E^N = \{T_{1.1.1}\}$$

Es ergibt sich somit:

$$E^D = \{G_1, G_{1.1}, B_{2.1.1}\}$$

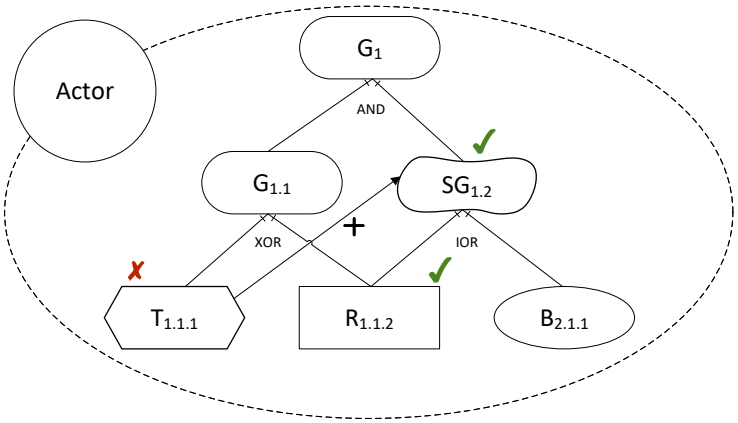


Abbildung 6.6: Zielauswahl

6.2.2 Generierung von Sichten auf das SoS-Modell

Durch die Generierung von Sichten auf das SoS-Modell lässt sich untersuchen, welche SoS-Konfigurationen eine Menge ausgewählter intentionaler Elemente im Zielmodell erfüllen kann. Dies vereinfacht die Prüfung, ob intentionale Elemente für die SoS-Konfigurationen spezifiziert sind, von denen sie erfüllt werden sollen.

Sichten auf das SoS-Modell werden durch eine Modelltransformation erstellt. Algorithmus 2 definiert die Erzeugung einer SoS-Konfigurationssicht, die aufzeigt, welche SoS-Konfigurationen alle ausgewählten intentionalen Elemente erfüllen können.

Als Input benötigt der Algorithmus das ursprüngliche SoS-Modell SM, die X-Links XL zwischen den Elementen des Zielmodells und des SoS-Modells sowie eine Auswahl von gewünschten intentionalen Elementen GM'. Als Ergebnis erzeugt der Algorithmus eine SoS-Konfigurationssicht SM', die restriktivere Kardinalitäten enthalten kann als das SoS-Modell.

Eine Iteration über alle Exclude-X-Links $exc \in EXC$ ermöglicht die Identifizierung der Excludes-X-Links, die mit den gewünschten intentionalen Elementen E^W verbunden sind. Um sicherzustellen, dass die SoS-Konfigurationen nicht die Anzahl von Systemen (oder Systemen dieser Kategorie) enthalten, die das ausgewählte intentionale Element ausschließt, werden die Differenzen zwischen den Kardinalitäten des Systemtyps/der Kategorie $i_{exc.sc}$ und den

Kardinalitäten der Excludes-X-Links $i_{exc.sc}$ berechnet und als neue Kardinalitäten für den Systemtyp/die Kategorie $i_{exc.sc}$ festgelegt. Die Constraints aus dem ursprünglichen SoS-Modell werden der Sicht hinzugefügt, da diese auch für die Sicht gelten [5]. Danach erfolgt im Rahmen einer Iteration über alle Requires-X-Links $req \in REQ$ die Identifizierung der Systemtypen oder Kategorien, die mit ausgewählten intentionalen Elementen E^W verbunden sind. Hier wird unterschieden, ob es sich um einen einfachen oder einen komplexen Requires-X-Link handelt. Bei einem komplexen Requires-X-Link wird der Sicht ein demgemäßes Requires-Constraint hinzugefügt. Bei einem einfachen Requires-X-Link müssen die Kardinalitäten des Systemtyps/der Systemkategorie $i_{req.sc}$ auf die Schnittmenge der Kardinalitäten des Systemtyps/der Systemkategorie $i_{req.sc}$ und der Kardinalitäten der Requires-X-Links i_{req} gesetzt werden, um sicherzustellen, dass nur die Anzahl der Systeme (oder der Systeme dieser Kategorie) Teil der SoS-Konfiguration ist, die von dem X-Link gefordert wird.

Algorithmus 2 Erstellung SoS-Konfigurationssicht

1: **for all** $exc \in EXC$ **do**
2: **if** $exc.e \in E^W$ **then** $i_{exc.sc} = i_{exc.sc} \setminus i_{exc}$
3: $CE' = CE$
4: **for all** $req \in REQ$ **do**
5: **if** $req.e \in E^W$ **then**
6: **if** $|req.sc| > 1$ **then** $CE'_R = CE'_R \cup \{\langle req.sc, i_{req}\rangle\}$
7: **else**
8: $i_{req.sc} = i_{req.sc} \cap i_{req}$

6.2.3 Beispielhafte Erstellung einer SoS-Konfigurationssicht

Die Abbildungen 6.7 und 6.8 illustrieren die Generierung einer SoS-Konfigurationssicht an dem in Kapitel 5 vorgestellten Beispiel der Transportroboterflotte. Abbildung 6.7 zeigt die gewünschten intentionalen Elemente im GRL-Zielmodell. Der Nutzer hat drei intentionale Elemente als gewünscht ausgewählt: die intentionalen Elemente *Warenübergabe auf variabler Höhe, Zusammenstöße vermeiden, Übergabe von Waren zwischen Robotern.* Die ausgewählten intentionalen Elemente sind in der Abbildung mit ✓ markiert. Abbildung 6.8 zeigt die generierte SoS-Konfigurationssicht für die gewünschten intentionalen Elemente aus Abbildung 6.7. Die Veränderungen der Sicht im Vergleich zu dem ursprünglichen Modell sind grün dargestellt. Die Sicht zeigt, dass die ausgewählten intentionalen Elemente nur erreichbar sind, wenn die Transportroboterflotte zwei bis zehn Roboter und entweder mindestens einen Roboter vom Typ FH oder vom Typ H enthält. In der Sicht ist ebenfalls zu erkennen, dass durch die Sichtengenerierung eine Inkonsistenz verursacht wurde. Obwohl die Transportrobotterflotte nur zehn Roboter enthalten darf, dürfen von den unterschiedlichen Typen von Robotern

[5] Entstandene Inkonsistenzen werden bei der in Abschnitt 7.2 vorgestellten Konsistenzprüfung von SoS-Modellen identifiziert

noch jeweils bis zu 20 Roboter Teil der Transportroboterflotte sein. Wie diese und andere Inkonsistenzen, die bei der Sichtengenerierung auftreten können, identifiziert und falls möglich automatisiert behoben werden können, wird im nächsten Kapitel erläutert.

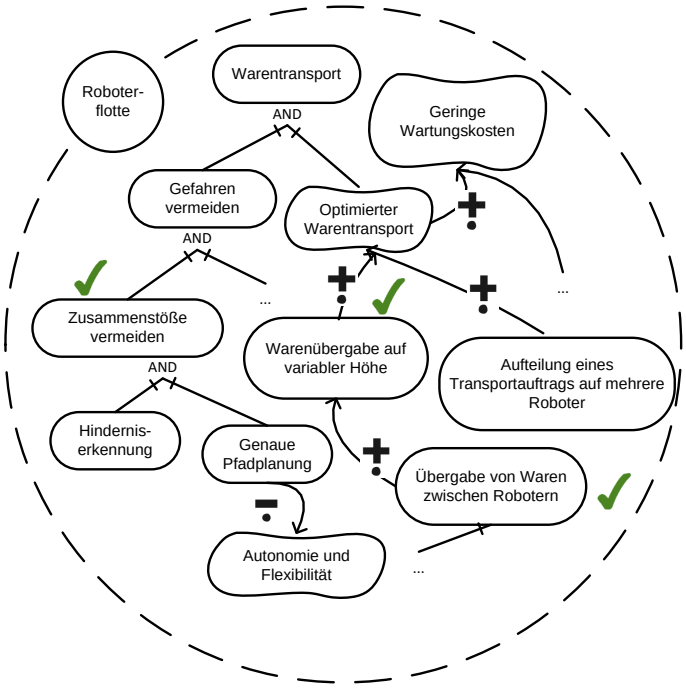


Abbildung 6.7: Beispiel für ausgewählte intentionale Elemente zur Generierung einer SoS-Konfigurationssicht

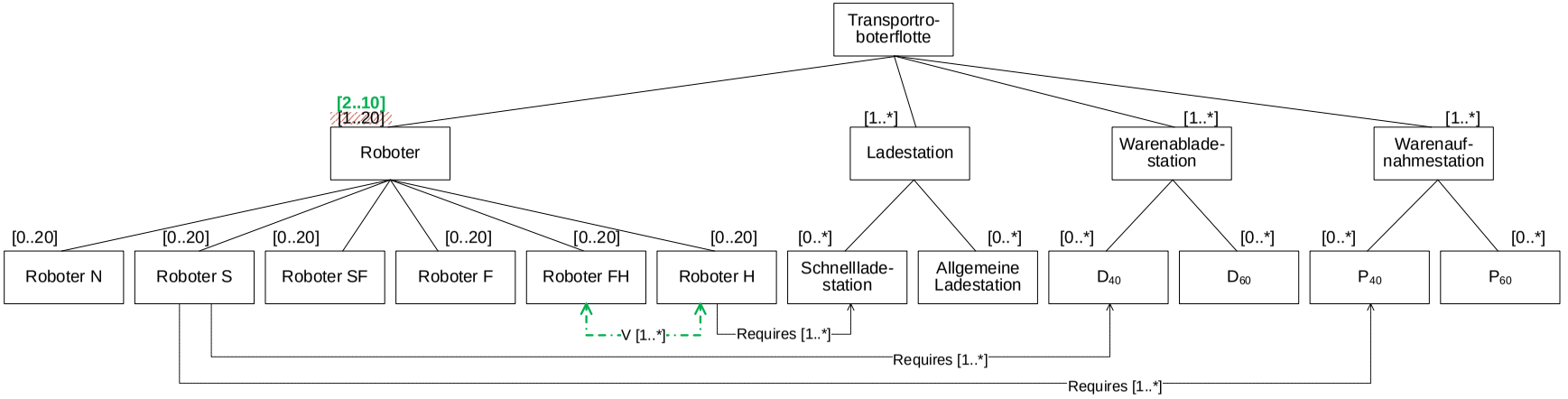


Abbildung 6.8: Beispiel für eine SoS-Konfigurationssicht basierend auf den ausgewählten intentionalen Elementen

7 Kapitel
Konsistenzprüfung bei der Sichtengenerierung

Im Rahmen der Generierung von Ziel- und SoS-Konfigurationssichten kann es zu Widersprüchen in der Auswahl von intentionalen Elementen bzw. SoS-Konfigurationen sowie in den Sichten kommen. Da sowohl eine widersprüchliche Auswahl von intentionalen Elementen bzw. SoS-Konfigurationen als auch Widersprüche in einer generierten Sicht zu ungültigen Sichten führen, ist deren Prüfung erforderlich.

Dieses Kapitel detailliert Schritt 2 des Lösungsansatzes. Es wird dargestellt, wie Ziel- und SoS-Modelle automatisiert geprüft werden, um ungültige Sichten zu identifizieren bzw. deren Erstellung zu verhindern.

Bei der Erstellung einer Zielsicht können bei der Auswahl der SoS-Konfigurationen Widersprüche auftreten. Beispielsweise könnte ausgewählt werden, dass ein System eines bestimmten Typs mindestens einmal Teil des System-of-Systems sein muss, obwohl Systeme dieser Kategorie ausgeschlossen wurden. Daher findet bei der Generierung von Zielsichten zunächst eine Prüfung statt, ob die Auswahl der SoS-Konfigurationen Inkonsistenzen enthält. Abbildung 7.1 zeigt, an welchen Stellen bei der Erstellung von Zielsichten Konsistenzprüfungen durchgeführt werden. Die möglichen Inkonsistenzen werden in Abschnitt 7.2 vorgestellt. Hier lassen sich automatisiert behebbare und nicht automatisiert behebbare Inkonsistenzen identifizieren. Sind nicht automatisiert behebbare Inkonsistenzen vorhanden, kann keine Zielsicht erstellt werden. Liegen ausschließlich automatisiert behebbare Inkonsistenzen vor, werden diese behoben, bevor die Zielsicht erstellt wird, wie Abschnitt 7.2.2 beschreibt. Nach der Erstellung der Zielsicht wird diese ebenfalls auf Inkonsistenzen geprüft. In der generierten Zielsicht können Widersprüche auftreten, wie beispielsweise, dass ein Oberelement einer AND-Dekomposition als erfüllbar identifiziert wurde, obwohl ein Unterelement als nicht erfüllbar identifiziert wurde. Die Inkonsistenzen, die in Zielsichten auftreten können, werden in Abschnitt 7.1.2 beschrieben. Enthält die Sicht nicht automatisiert behebbare Inkonsistenzen, liegt entweder ein Defekt in der SoS-Zielspezifikation vor oder die SoS-Konfigurationen wurden so gewählt, dass widersprüchliche intentionale Elemente erfüllbar sind. Enthält die Zielsicht keine nicht automatisiert

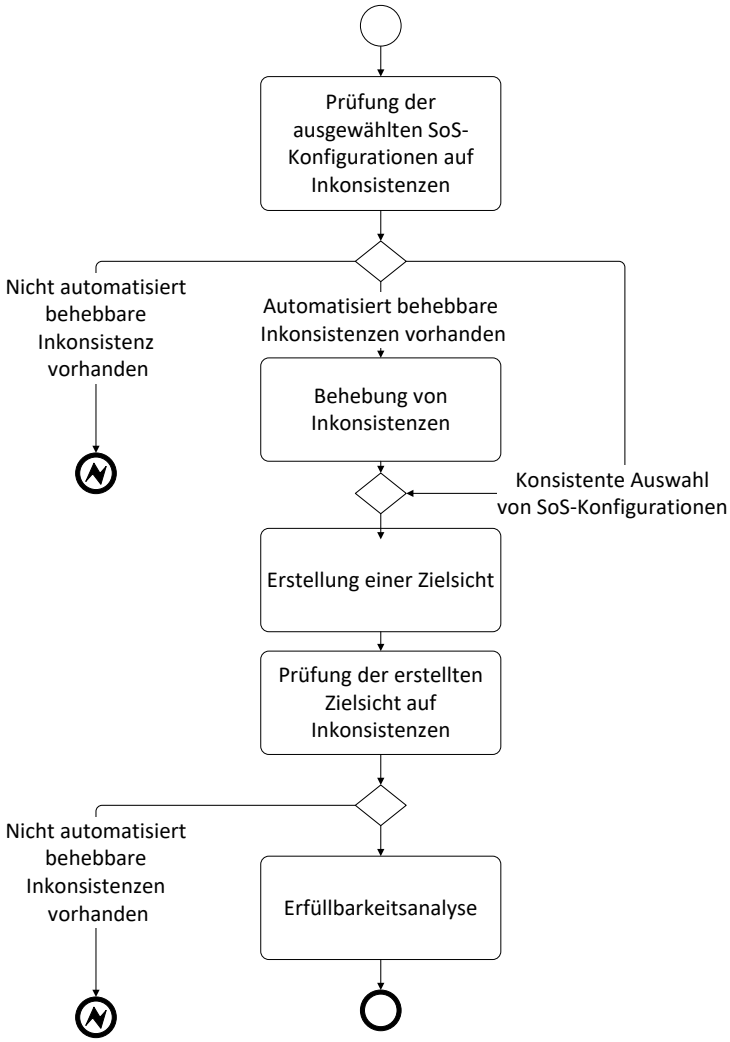


Abbildung 7.1: Erstellung einer Zielsicht mit Konsistenzprüfungen

behebbaren Inkonsistenzen, wird eine Erfüllbarkeitsanalyse gemäß den in Abschnitt 7.1.1 beschriebenen Erfüllbarkeitsregeln für GRL-Zielmodelle durchgeführt, um weitere erfüllbare bzw. nicht erfüllbare intentionale Elemente in der Zielsicht zu identifizieren.

Bei der Erstellung einer SoS-Konfigurationssicht kann es bei der Auswahl der gewünschten intentionalen Elementen zu Widersprüchen kommen. Beispielsweise könnte ausgewählt sein, dass ein Oberelement einer AND-Dekomposition gewünscht ist, obwohl ein Unterelement nicht gewünscht ist. Daher findet bei der Generierung von SoS-Konfigurationssichten zunächst eine Prüfung statt, ob die Auswahl der intentionalen Elemente Inkonsistenzen enthält. Abbildung 7.2 zeigt, an welchen Stellen bei der Erstellung von SoS-Konfigurationssichten Konsistenzprüfungen durchgeführt werden. Die möglichen Inkonsistenzen sind in Abschnitt 7.1.2 beschrieben. Hierbei können nicht automatisiert behebbare Inkonsistenzen und eine

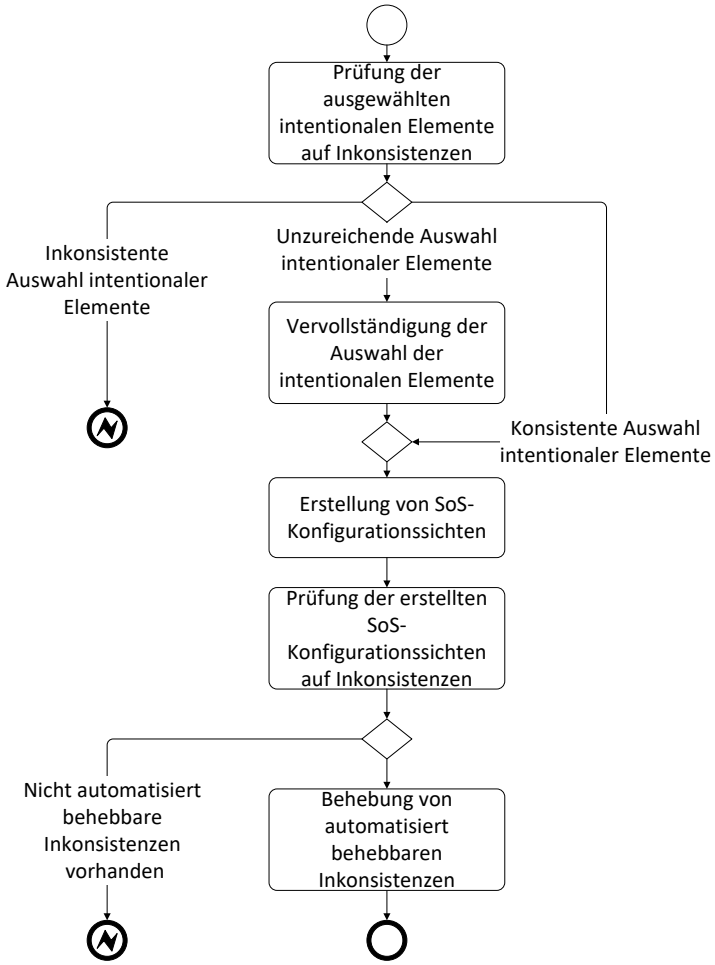


Abbildung 7.2: Erstellung einer SoS-Konfigurationssicht mit Konsistenzprüfungen

unzureichende Auswahl intentionaler Elemente identifiziert werden. Sind nicht automatisiert behebbare Inkonsistenzen vorhanden, kann keine SoS-Konfigurationssicht erstellt werden. Liegt nur eine unzureichende Auswahl intentionaler Elemente vor, werden diese behoben, bevor die SoS-Konfigurationssicht erstellt wird, wie Abschnitt 7.1.4 beschreibt.

In der generierten SoS-Konfigurationssicht können ebenfalls Widersprüche auftreten, wie beispielsweise, dass ein System eines bestimmten Typs mindestens einmal Teil des System-of-Systems sein soll, obwohl Systeme dieser Kategorie nicht Teil des System-of-Systems sind. Die Inkonsistenzen, die in SoS-Konfigurationssichten auftreten können, werden in Abschnitt 7.2 beschrieben. Enthält die Sicht nicht automatisiert behebbare Inkonsistenzen, liegt entweder ein Defekt in der SoS-Zielspezifikation vor oder die intentionalen Elemente wurden so gewählt, dass sie für keine der spezifizierten SoS-Konfigurationen erfüllbar sind. Enthält die SoS-Konfigurationssicht keine nicht automatisiert behebbaren Inkonsistenzen, werden eventuell vorhandene automatisiert behebbare Inkonsistenzen behoben.

Inkonsistenzen zwischen dem Zielmodell und dem SoS-Modell werden bei der Sichtengenerierung erkannt. So lassen sich beispielsweise intentionale Elemente im Zielmodell identifizieren, die in keiner SoS-Konfiguration erfüllbar sind, da bei Auswahl des intentionalen Elements keine gültige SoS-Konfigurationssicht erstellt werden kann. Ebenfalls lassen sich SoS-Konfigurationen ohne erfüllbare intentionale Elemente bei dem Versuch der Erstellung einer diesbezüglichen Zielsicht identifizieren. Für eine automatisierte Analyse der Beziehungen zwischen Ziel- und SoS-Modell kann bereits vor der Generierung von Sichten der Ansatz von Metzger et al. (2007) genutzt werden, um alle intentionalen Elemente zu identifizieren, die für keine SoS-Konfiguration erfüllbar sind, sowie um alle SoS-Konfigurationen zu erkennen, für die keine intentionalen Elemente erfüllbar sind.

7.1 Konsistenzprüfung von Zielmodellen

Dieser Abschnitt beschreibt die Konsistenzprüfung von Zielmodellen, die zur Anwendung kommt, um die Auswahl der intentionalen Elemente im Zielmodell und die Zielsicht auf Inkonsistenzen zu prüfen. Sind Inkonsistenzen vorhanden, kann keine gültige Sicht generiert werden. Abschnitt 7.1.1 stellt die Erfüllbarkeitsregeln für GRL-Zielmodelle vor, die genutzt werden, um Zielmodelle auf Konsistenz zu prüfen. Abschnitt 7.1.2 befasst sich mit den Inkonsistenzen, die in der Auswahl der intentionalen Elemente im Zielmodell und der Zielsicht auftreten können. Abschnitt 7.1.3 illustriert mögliche Inkonsistenzen an einem Beispiel. Abschnitt 7.1.4 zeigt, wie die Erfüllbarkeitsregeln genutzt werden, um bei einer konsistenten, aber unzureichenden Auswahl von intentionalen Elementen im Zielmodell alle relevanten Kombinationen zu ermitteln.

7.1.1 Erfüllbarkeitsregeln für GRL-Zielmodelle

Die Konsistenzprüfung der Auswahl der intentionalen Elemente und der Zielsicht basiert auf der in Abschnitt 2.2.2 vorgestellten Erfüllbarkeitsprüfung von GRL-Zielmodellen von Amyot et al. (2010).

Es werden zwei Erfüllbarkeitsgrade differenziert und durch die folgenden Prädikate beschrieben: $S(e)$ bedeutet, dass Element e erfüllbar ist; $D(e)$ bedeutet, dass Element e nicht erfüllbar ist[6].

[6] Im Rahmen der Auswahl der intentionalen Elemente wird nicht zwischen erfüllbaren und nicht erfüllbaren Elementen, sondern zwischen gewünschten und nicht gewünschten Elementen unterschieden. Da die Konsistenzregeln analog gelten, werden zur Vermeidung von Wiederholungen bei der Konsistenzprüfung der Auswahl der intentionalen Elemente gewünschte Elemente als erfüllbare und nicht gewünschte als nicht erfüllbare Elemente angesehen. Der Erfüllungsgrad teilweise erfüllbar wird hier nicht betrachtet. Bei der Auswahl der intentionalen Elemente kann nur zwischen gewünscht und nicht gewünscht entschieden werden. Bei einer Zielsicht bedeutet teilweise erfüllbar, dass das intentionale Element für einen Teil der ausgewählten SoS-Konfigurationen erfüllbar ist, aber nicht für alle ausgewählten. Daher kann ein teilweise erfüllbares intentionales Element weder im Widerspruch zu einem erfüllbaren noch zu einem nicht erfüllbaren intentionalen Element stehen.

Ein intentionales Element ist immer entweder erfüllbar oder nicht erfüllbar. Daher gilt:

$$S(e) \Leftrightarrow \neg D(e)$$

$$D(e) \Leftrightarrow \neg S(e)$$

Die Erfüllbarkeit von intentionalen Elementen $e \in E$ ergibt sich aus ihren Verbindungen zu anderen Elementen und deren Erfüllbarkeit.

Für eine UND-Dekomposition $\langle e_O, E_U \rangle$ gilt, dass das Oberelement e_O erfüllbar ist, wenn alle Unterelemente $E_U = \{e_1, ..., e_n\}$ erfüllbar sind. Das heißt:

$$(\bigwedge_{i=1}^{n} S(e_i)) \rightarrow S(e_O)$$

Ist mindestens ein Unterelement nicht erfüllbar, ist das Oberelement ebenfalls nicht erfüllbar.

$$(\bigvee_{i=1}^{n} D(e_i)) \rightarrow D(e_O)$$

Für eine IOR-Dekomposition $\langle e_O, E_U \rangle$ gilt, dass das Oberelement e_O erfüllbar ist, wenn mindestens ein Unterelement $E_U = \{e_1, ..., e_n\}$ erfüllbar ist. Das heißt:

$$(\bigvee_{i=1}^{n} S(e_i)) \rightarrow S(e_O)$$

Ist kein Unterelement erfüllbar, ist das Oberelement ebenfalls nicht erfüllbar.

$$(\bigwedge_{i=1}^{n} D(e_i)) \rightarrow D(e_O)$$

Für eine XOR-Dekomposition $\langle e_O, E_U \rangle$ gilt, dass das Oberelement e_O erfüllbar ist, wenn genau ein Unterelement $E_U = \{e_1, ..., e_n\}$ erfüllbar ist. Das heißt:

$$(\exists(\bigvee_{i=1}^{n} S(e_i)) \wedge \nexists(\bigvee_{j=1}^{n} S(e_j)) \wedge i \neq j) \rightarrow S(e_O)$$

Ist kein oder mehr als ein Unterelement erfüllbar, ist das Oberelement ebenfalls nicht erfüllbar.

$$(\exists(\bigvee_{i=1}^{n} S(e_i)) \wedge \nexists(\bigvee_{j=1}^{n} S(e_j)) \wedge i \neq j \vee \bigwedge_{i=1}^{n} D(e_i)) \rightarrow D(e_O)$$

Für Contributions gilt, dass das unterstützte Element e erfüllbar ist, wenn stärkere positive Einflüsse darauf einwirken als negative. Dazu wird für alle eingehenden Contributions der Erfüllungsgrad des unterstützenden Elements mit dem Wert der jeweiligen Contribution multipliziert und aufsummiert. Ist dieser Wert größer als 0, ist e erfüllbar. Ist dieser Wert kleiner oder gleich 0, ist e nicht erfüllbar.

$$\sum_{\forall\langle e_i,e,v_i\rangle\in Con} (v_i \times x(e_i)) > 0) \rightarrow S(e)$$

$$\sum_{\forall\langle e_i,e,v_i\rangle\in Con} (v_i \times x(e_i)) \leq 0) \rightarrow D(e)$$

Wobei,

$$x(e) = \begin{cases} 1 & \text{für } S(e) \\ -1 & \text{sonst} \end{cases}$$

Kommen qualitative Contributionwerte zum Einsatz, werden diese wie bei dem hybriden Erfüllbarkeitsprüfungsansatz (vgl. Abschnitt 2.2.2.3 in ganzzahlige Werte überführt.

Für eine Dependency gilt, dass das abhängige intentionale Element i_{d2} den Erfüllbarkeitsgrad des intentionalen Elements i_{d1} annimmt. Das heißt:

$$S(i_{d1}) \rightarrow S(i_{d2})$$

$$D(i_{d1}) \rightarrow D(i_{d2})$$

7.1.2 Inkonsistenzen in GRL-Zielmodellen bezogen auf die Erfüllbarkeit

Basierend auf den vorgestellten Regeln für die Erfüllbarkeitsprüfungen ergeben sich für Zielmodelle die folgenden Wohlgeformtheitsregeln, die bei der Auswahl der intentionalen Elemente und bei Zielsichten überprüft werden müssen, um die Erstellung fehlerhafter Sichten zu verhindern.

Tabelle 7.1 zeigt die möglichen Inkonsistenzen bei UND-Dekompositionen. Da ein Oberelement in einer UND-Dekomposition nur erfüllbar ist, wenn alle Unterelemente erfüllbar sind, können hier zwei unterschiedliche Fälle auftreten. Erstens ist das Oberelement erfüllbar, aber mindestens ein Unterelement ist nicht erfüllbar. Zweitens sind alle Unterelemente erfüllbar, aber das Oberelement nicht.

Tabelle 7.2 zeigt die möglichen Inkonsistenzen bei IOR-Dekompositionen. Da ein Oberelement in einer IOR-Dekomposition nur erfüllbar ist, wenn mindestens ein Unterelement erfüllbar ist, können hier zwei unterschiedliche Fälle auftreten. Erstens ist das Oberelement erfüllbar, aber kein Unterelement ist erfüllbar. Zweitens ist mindestens ein Unterelement erfüllbar, aber das Oberelement nicht.

Tabelle 7.3 zeigt die möglichen Inkonsistenzen bei XOR-Dekompositionen. Da ein Oberelement in einer XOR-Dekomposition nur erfüllbar ist, wenn genau ein Unterelement erfüllbar ist, können hier drei unterschiedliche Fälle auftreten. Erstens ist das Oberelement erfüllbar, aber kein Unterelement ist erfüllbar. Zweitens ist das Oberelement erfüllbar, aber mehr als ein Unterelement ist erfüllbar. Drittens ist genau ein Unterelement erfüllbar, aber das Oberelement nicht.

Tabelle 7.1: Inkonsistenzen bei UND-Dekompositionen

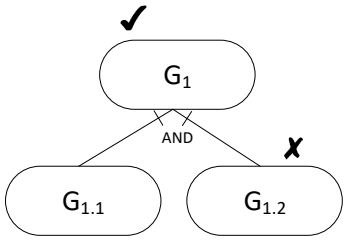
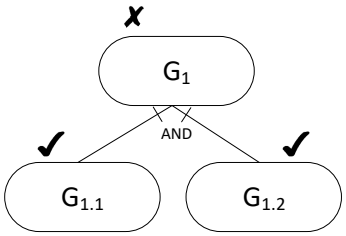
Oberziel erfüllbar, ein oder mehr Unterziele nicht erfüllbar $\exists \langle e, E_U \rangle \in Decomp^{AND} : S(e) \wedge e.sub \in E_U \wedge D(e.sub)$	
Oberziel nicht erfüllbar, alle Unterziele erfüllbar $\exists \langle e, E_U \rangle \in Decomp^{AND} : (D(e) \wedge \nexists e.sub \in E_U : D(e.sub))$	

Tabelle 7.2: Inkonsistenzen bei IOR-Dekompositionen

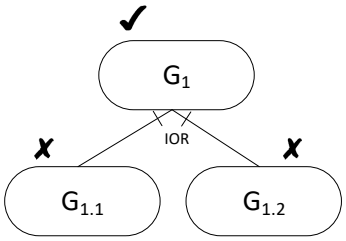
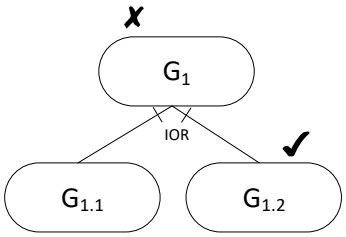
Oberziel erfüllbar, alle Unterziele nicht erfüllbar $\exists \langle e, E_U \rangle \in Decomp^{IOR} : (S(e) \wedge \nexists e.sub \in E_U : \neg D(e.sub)$	
Oberziel nicht erfüllbar, ein oder mehr Unterziele erfüllbar $\exists \langle e, E_U \rangle \in Decomp^{IOR} : D(e) \wedge e.sub \in E_U \wedge S(e.sub)$	

Tabelle 7.3: Inkonsistenzen bei XOR-Dekompositionen

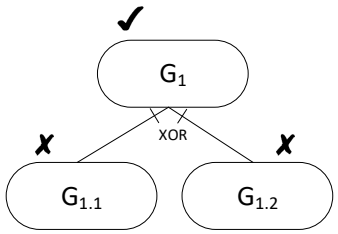
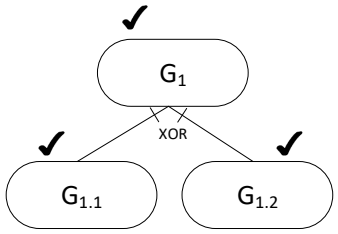
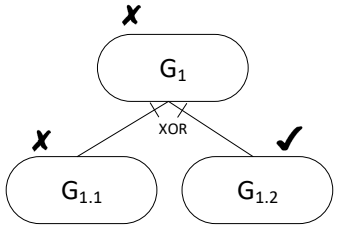
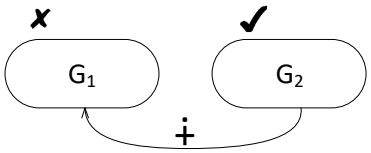
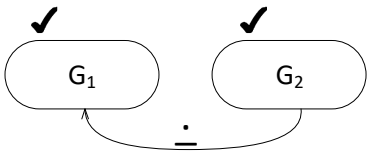
Oberziel erfüllbar, alle Unterziele nicht erfüllbar $\exists\langle e, E_U\rangle \in Decmp^{XOR} : S(e) \wedge e.sub \in E_U \wedge D(e.sub)$	
Oberziel erfüllbar, mehr als ein Unterziel erfüllbar $\exists\langle e, E_U\rangle \in Decmp^{XOR} : (e_1, e_2 \in E_U \wedge S(e) \wedge S(e_1) \wedge S(e_2))$	
Oberziel nicht erfüllbar, genau ein Unterziel erfüllbar $\exists\langle e, E_U\rangle \in Decmp^{XOR} : (S(e) \wedge \exists!e.sub \in E_U : (S(e.sub))$	

Tabelle 7.4 zeigt die möglichen Inkonsistenzen bei Contributions. Hier müssen Make und Break Contributions betrachtet werden. Make Contributions definieren, dass das unterstützte Element erfüllbar ist, wenn das unterstützende Element erfüllbar ist. Daraus ergibt sich, dass eine Inkonsistenz vorliegt, wenn bei einer Make Contribution dass unterstützende Element erfüllbar, das unterstützte Element aber nicht erfüllbar ist. Break Contributions definieren, dass das unterstützte Element nicht erfüllbar ist, wenn das unterstützende Element erfüllbar ist. Daraus ergibt sich, dass eine Inkonsistenz vorliegt, wenn bei einer Break Contribution das unterstützende Element erfüllbar, das unterstützte Element aber auch erfüllbar ist.

Tabelle 7.4: Inkonsistenzen bei Contributions

Unterstützendes Ziel erfüllbar, unterstütztes Ziel nicht erfüllbar bei Make-Contribution	
$\exists\langle e_1, e_2, Make\rangle \in Con : S(e_1) \wedge D(e_2)$	
Unterstützendes Ziel erfüllbar, unterstütztes Ziel erfüllbar bei Break-Contribution	
$\exists\langle e_1, e_2, Break\rangle \in Con : S(e_1) \wedge S(e_2)$	

7.1.3 Beispiel für Inkonsistenzen in Zielmodellen

Abbildung 7.3 zeigt eine Inkonsistenz in dem Zielmodell der Transportroboterflotte. Das Ziel *Zusammenstöße vermeiden* ist als erfüllbar gekennzeichnet. Dieses Ziel wird durch eine UND-Dekomposition in die Ziele *Hinderniserkennung* und genaue *Pfadplanung* verfeinert. Das Ziel *Zusammenstöße vermeiden* ist nur dann erfüllbar, wenn auch alle Unterziele erfüllbar sind. Allerdings ist hier *Hinderniserkennung* nicht erfüllbar. Es liegt somit eine Inkonsistenz vor. Für die Auswahl der intentionalen Elemente bedeutet dies, dass keine gültige SoS-Konfigurationssicht erzeugt werden kann, wenn das Ziel *Zusammenstöße vermeiden* gewünscht und das Ziel *Hinderniserkennung* nicht gewünscht ist. Liegt diese Inkonsistenz in einer erzeugten Zielsicht vor, existiert für die gewünschten SoS-Konfigurationen auch keine gültige Zielsicht.

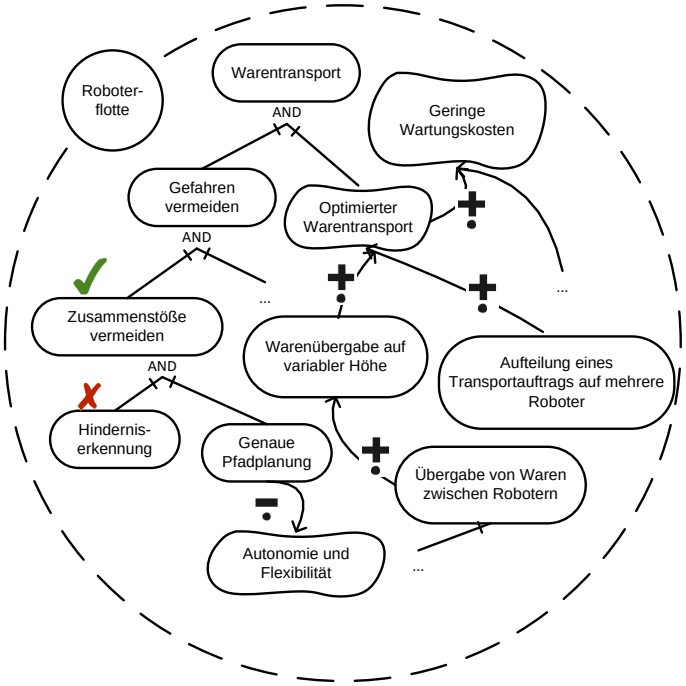


Abbildung 7.3: Inkonsistentes Zielmodell

7.1.4 Unzureichende Auswahl der intentionalen Elemente

Aufgrund der Erfüllbarkeitregeln für GRL-Zielmodelle ergeben sich Abhängigkeiten, die indirekt Auswirkungen auf die SoS-Konfigurationssicht haben können. Abbildung 7.4 illustriert, welche Auswirkungen eine unzureichende Auswahl der intentionalen Elemente auf die Bildung einer SoS-Konfigurationssicht haben kann. Wenn der Nutzer intentionale Elemente als gewünscht auswählt, wie hier in dem Beispiel das Ziel G_1, die in Unterziele (hier $G_{1.1}$ und $SG_{1.2}$) dekomponiert sind, und diese Unterziele vom Nutzer weder als gewünscht noch als ungewünscht eingeordnet wurden, ergeben sich verschiedene mögliche Kombinationen für die Erfüllbarkeitsgrade der Unterziele (hier $G_{1.1}$ gewünscht und $SG_{1.2}$ nicht gewünscht oder $G_{1.1}$ nicht gewünscht und $SG_{1.2}$ gewünscht). Diese Kombinationen können durch Backward Reasoning automatisiert ermittelt werden (Horkoff und Yu 2010).

Für das Backward Reasoning müssen die verschiedenen Arten von Dekompositionen und Make-Contributions berücksichtigt werden. Im Folgenden werden die Regeln für das Backward Reasoning an Minimalbeispielen illustriert. Zur Unterscheidung von manuell gewünschten und automatisiert ermittelten Belegungen sind die ermittelten Belegungen jeweils in Klammern dargestellt.

Tabelle 7.5 zeigt, welche Auswirkungen ein gewünschtes Oberziel einer UND-Dekomposition hat. Wenn das Oberziel gewünscht ist, gelten auch alle Unterziele als gewünscht.

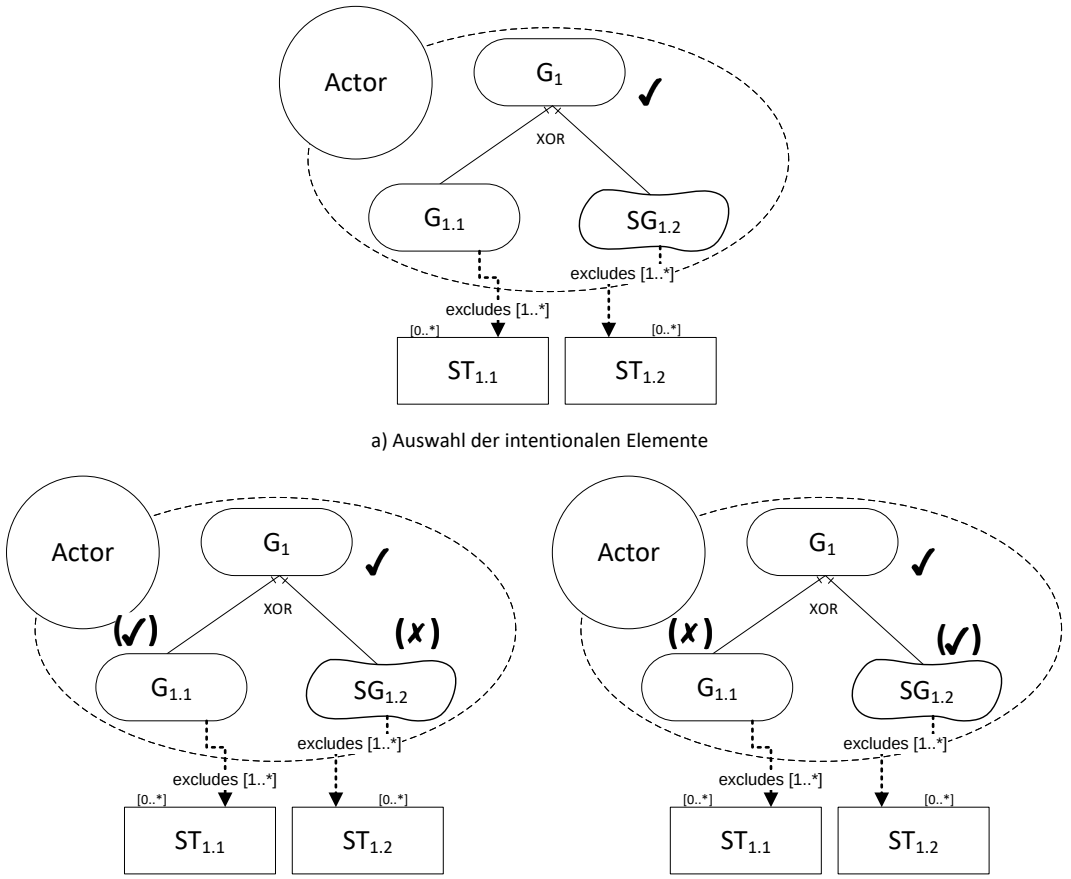


Abbildung 7.4: Unzureichende Auswahl der intentionalen Elemente

Tabelle 7.5: Backward-Reasoning bei UND-Dekompositionen

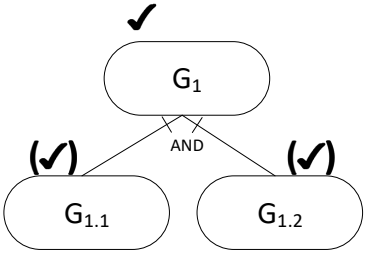
Oberziel gewünscht führt zu alle Unterziele gewünscht. $(\forall\langle e, E_U\rangle \in Decomp^{AND} \vert e \in E^W) \implies (\nexists e.sub \in E_U \vert e.sub \in E^N)$	

Tabelle 7.6 zeigt, welche Auswirkungen ein gewünschtes Oberziel einer IOR-Dekomposition hat. Wenn das Oberziel gewünscht ist, gilt mindestens ein Unterziel als gewünscht. Für Oberziele mit zwei Unterzielen ergeben sich die drei dargestellten Möglichkeiten: beide Unterziele gewünscht, das erste Unterziel gewünscht und das zweite Unterziel nicht gewünscht oder das erste Unterziel nicht gewünscht und das zweite Unterziel gewünscht.

Tabelle 7.6: Backward-Reasoning bei IOR-Dekompositionen

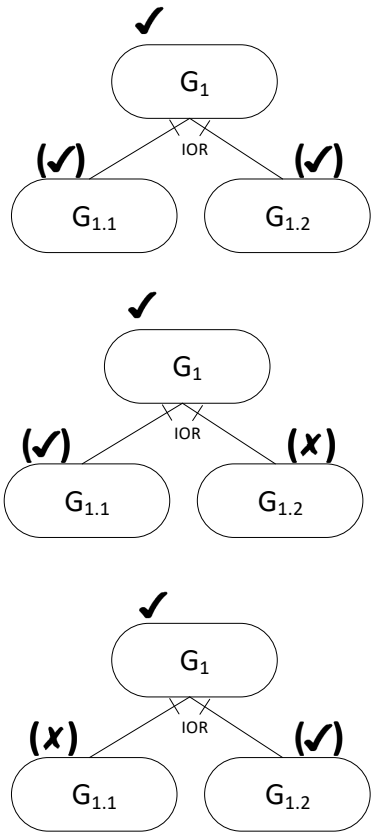
Oberziel gewünscht führt zu mindestens einem Unterziele gewünscht. $(\forall\langle e, E_U\rangle \in Decomp^{IOR} \vert e \in E^W) \implies (\exists e.sub \in E_U \vert e.sub \in E^W)$	

Tabelle 7.7 zeigt, welche Auswirkungen ein gewünschtes Oberziel einer XOR-Dekomposition hat. Wenn das Oberziel gewünscht ist, gilt mindestens ein Unterziel als gewünscht. Für Oberziele mit zwei Unterzielen ergeben sich die zwei dargestellten Möglichkeiten: das erste Unterziel gewünscht und das zweite Unterziel nicht gewünscht oder das erste Unterziel nicht gewünscht und das zweite Unterziel gewünscht.

Tabelle 7.8 verdeutlicht, welche Auswirkungen ein gewünschtes unterstützendes Ziel einer Make-Contribution hat. Wenn das unterstützende Ziel gewünscht ist, gilt das unterstützte Ziel als gewünscht.

Das Backwards Reasoning findet nach der Auswahl der intentionalen Elemente durch den Nutzer statt. Dabei können Inkonsistenzen entstehen, wie sie in Abschnitt 7.1.2 beschrieben wurden. Dies führt dazu, dass keine gültige SoS-Konfigurationssicht erstellt werden kann.

Tabelle 7.7: Backward-Reasoning bei XOR-Dekompositionen

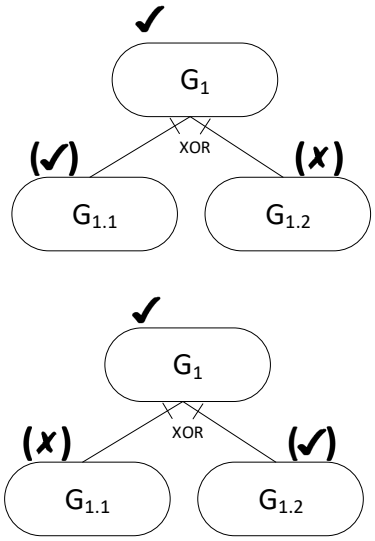
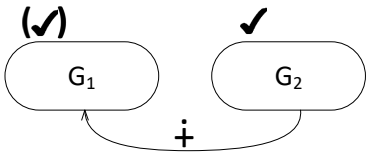
Oberziel gewünscht führt zu genau einem Unterziele gewünscht. $(\forall\langle e, E_U\rangle \in Decomp^{XOR} \vert e \in E^W) \implies (\exists! e.sub \in E_U \vert e.sub \in E^W)$	

Tabelle 7.8: Backward-Reasoning bei Make-Contributions

Unterstützendes Ziel gewünscht führt zu unterstütztes Ziel gewünscht. $(\forall\langle e_1, e_2, Make\rangle \vert e_1 \in E^W) \implies (e_2 \in E^W)$	

Das Backwards Reasoning kann zu einer großen Anzahl von unterschiedlichen Mengen an gewählten intentionalen Elementen führen, die wiederum die Generierung einer Vielzahl von SoS-Konfigurationssichten basierend auf nur einer ursprünglichen Nutzerauswahl nach sich ziehen kann. Allerdings ist die Unterscheidung zwischen gewünschten und nicht gewünschten Elementen nur von Interesse, falls das Element durch einen X-Link mit dem SoS-Modell verbunden ist. Daher müssen nur die Mengen an gewählten intentionalen Elementen bei der Generierung von SoS-Konfigurationssichten berücksichtigt werden, die auch zu unterschiedlichen SoS-Konfigurationssichten führen. Für jede relevante Menge an gewählten intentionalen Elementen wird eine SoS-Konfigurationssicht generiert.

7.2 Konsistenzprüfung von SoS-Modellen

Die Konsistenzprüfung der Auswahl der SoS-Konfigurationen und der SoS-Konfigurationssicht basiert auf der in Schnabel et al. (2016); Weckesser et al. (2016) beschriebenen Analyse von kardinalitätsbasierten Feature-Modellen. In kardinalitätsbasierten Feature-Modellen lassen sich Inkonsistenzen auf Widersprüche zwischen Kardinalitäten zurückführen. Hierbei lassen sich zwei Arten von Widersprüchen unterscheiden. Widersprüche, die keine gültigen SoS-Konfigurationen zulassen, sind nicht automatisiert behebbar und deuten auf Fehler in der Spezifikation oder in der Auswahl hin. Dies tritt beispielsweise auf, wenn spezifiziert ist, dass SoS-Konfigurationen mindestens sechs *Roboter vom Typ H*, aber maximal fünf *Roboter* enthalten müssen. Manche Widersprüche sind automatisiert behebbar. In diesen Fällen kommt es dazu, dass Kardinalitäten fälschlicherweise suggerieren, dass mehr SoS-Konfigurationen zulässig sind als tatsächlich möglich. Dies tritt beispielsweise auf, wenn spezifiziert ist, dass mindestens fünf *Roboter* enthalten sein müssen, wenn auch spezifiziert ist, dass mindestens sechs *Roboter vom Typ H* enthalten sein müssen. Dieser Widerspruch kann behoben werden, indem die Minimalkardinalität für Roboter erhöht wird.

7.2.1 Nicht automatisiert behebbare Inkonsistenzen in SoS-Modellen

Nicht automatisiert behebbare Inkonsistenzen in SoS-Modellen können prinzipiell an zwei Stellen auftreten:

1. zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen
2. zwischen den Kardinalitäten von Constraintkanten und Kategorien bzw. Systemtypen

7.2.1.1 Nicht automatisiert behebbare Inkonsistenzen zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen

Die verschiedenen Fälle von Inkonsistenzen zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen werden in diesem und den folgenden Abschnitten

anhand des Beispiels aus Abbildung 7.5 illustriert. Die Abbildung zeigt einen vereinfachten Ausschnitt aus einem SoS-Modell mit einer Kategorie und drei Systemtypen. Prinzipiell können die vorgestellten Inkonsistenzen auch bei einer anderen Anzahl von Systemtypen sowie zwischen Unter- und Oberkategorien auftreten.

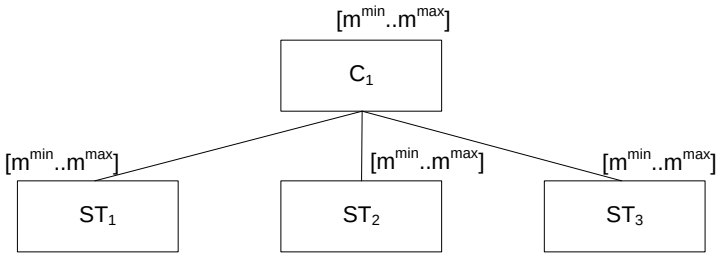
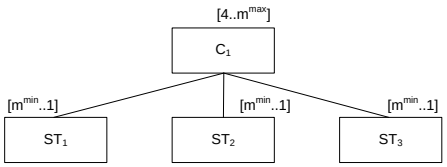
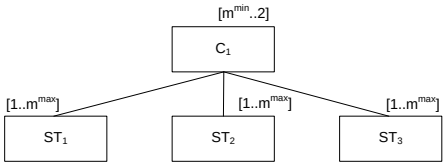


Abbildung 7.5: Inkonsistenzen in SoS-Modellen

Bei nicht automatisiert behebbaren Inkonsistenzen kommt es zu Widersprüchen zwischen den Kardinalitäten der Oberkategorie und der Unterkategorie bzw. Systemtypen. Anders als bei automatisiert behebbaren Inkonsistenzen schließen sich in diesem Fall die Kardinalitäten gegenseitig aus. Das SoS-Modell zeigt dementsprechend keine gültigen SoS-Konfigurationen. Hier können zwei Fälle auftreten, die betrachtet werden müssen.

Tabelle 7.9: Nicht automatisiert behebbare Inkonsistenzen

Minimalkardinalität der Oberkategorie größer als Summe der Maximalkardinalitäten der Systemtypen $m^{min}(C_1) > \sum_{n=1}^{i} m^{max}(ST_i)$	
Maximalkardinalität der Oberkategorie kleiner als Summe der Minimalkardinalität der Systemtypen $m^{max}(C_1) < \sum_{n=1}^{i} m^{min}(ST_i)$	

Im ersten Fall ist die Minimalkardinalität m^{min} der Oberkategorie C_1 größer als die Summe aller Maximalkardinalitäten der Unterkategorien bzw. Systemtypen ST_i. Das Beispiel in Tabelle 7.9 zeigt, dass mindestens vier Systeme der Kategorie C_1 Teil der SoS-Konfiguration sein sollen. Allerdings gehören zu der Kategorie C_1 nur drei Systemtypen, von denen jeweils maximal ein System Teil der SoS-Konfiguration sein soll. Dies führt dazu, dass das SoS-Modell keine

gültigen SoS-Konfigurationen spezifiziert. Bei einem angepassten SoS-Modell für die Auswahl der SoS-Konfigurationen liegt somit eine ungültige Auswahl von SoS-Konfigurationen vor, für die keine Zielsicht erstellt werden kann. Bei der SoS-Konfigurationssicht handelt es sich um eine ungültige Sicht, was bedeutet, dass die ausgewählten Ziele von keiner SoS-Konfiguration erfüllt werden können.

Im zweiten Fall ist die Maximalkardinalität m^{max} der Oberkategorie C_1 kleiner als die Summe aller Minimalkardinalitäten der Unterkategorien bzw. Systemtypen ST_i. Das Beispiel in Tabelle 7.9 zeigt, dass maximal zwei Systeme der Kategorie C_1 Teil der SoS-Konfiguration sein sollen. Allerdings gehören zu der Kategorie C_1 drei Systemtypen, von denen jeweils mindestens ein System Teil der SoS-Konfiguration sein soll. Dies führt ebenfalls dazu, dass das SoS-Modell keine gültigen SoS-Konfigurationen spezifiziert. Bei einem angepassten SoS-Modell für die Auswahl der SoS-Konfigurationen kommt es auch hier zu einer ungültigen Auswahl von SoS-Konfigurationen, für die keine Zielsicht erstellt werden kann. Bei der SoS-Konfigurationssicht handelt es sich ebenfalls um eine ungültige Sicht, was bedeutet, dass die ausgewählten Ziele von keiner SoS-Konfiguration erfüllt werden können.

7.2.1.2 Nicht automatisiert behebbare Inkonsistenzen zwischen den Kardinalitäten von Constraintkanten und Kategorien bzw. Systemtypen

Inkonsistenzen können ebenfalls durch die Verletzung von Constraints auftreten. Da im SoS-Modell Constraintkanten ebenfalls Kardinalitäten besitzen, müssen nicht nur die Art der Constraintkante, sondern auch die Kardinalitäten der Constraintkante und der betroffenen Knoten berücksichtigt werden.

Die verschiedenen Fälle von Inkonsistenzen zwischen den Kardinalitäten von Requires-Constraints und Kategorien bzw. Systemtypen werden in diesem und den folgenden Abschnitten anhand des Beispiels aus Abbildung 7.6 illustriert. Die Abbildung zeigt einen Ausschnitt aus einem SoS-Modell mit zwei Systemtypen und einem Requires-Constraint.

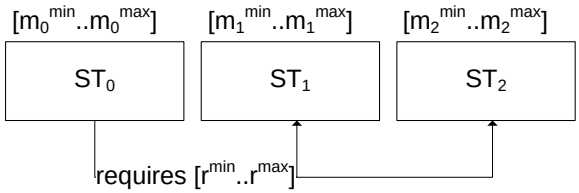


Abbildung 7.6: Inkonsistenzen bei Requires-Constraints

Eine nicht automatisiert behebbare Inkonsistenz liegt vor, wenn die Minimalkardinalität des Knotens ST_0, von dem die Requires-Constraintkante ausgeht, größer null ist und die Summe der Minimalkardinalitäten m_i^{min} der Knoten, auf die die Kante zeigt, größer als die Maximalkardinalität der Constraintkante r^{max} oder die Summe der Maximalkardinalitäten m_i^{max} der

Knoten, auf die die Kante zeigt, größer als die Minimalkardinalität der Constraintkante r^{min} ist.

Das heißt, wenn

$$m_0^{min} > 0 \quad \wedge \quad (\sum_{n=1}^{i} m_i^{min} > r^{max} \quad \vee \quad \sum_{n=1}^{i} m_i^{max} < r^{min})$$

In diesem Fall lassen sich keine gültigen SoS-Konfigurationen aus dem SoS-Modell ableiten.

Die verschiedenen Fälle von Inkonsistenzen zwischen den Kardinalitäten von Excludes-Constraints und Kategorien bzw. Systemtypen werden in diesem und den folgenden Abschnitten anhand des Beispiels aus Abbildung 7.7 illustriert. Die Abbildung zeigt einen Ausschnitt aus einem SoS-Modell mit zwei Systemtypen und einem Exclude-Constraint.

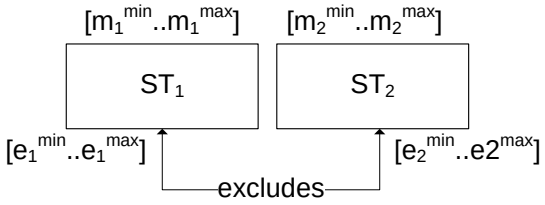


Abbildung 7.7: Inkonsistenzen bei Exclude-Constraints

Eine Inkonsistenz tritt auf, wenn bei beiden Knoten die Kardinalität des Knotens eine Teilmenge der Kardinalität der Exclude-Constraintkante ist. Dies führt dazu, dass alle SoS-Konfigurationen jeweils eine Anzahl von diesbezüglichen Systemen enthalten müssen, die durch die Kante ausgeschlossen werden. Das heißt, wenn

$$[m_1^{min}..m_1^{max}] \subseteq [e_1^{min}..e_1^{max}] \quad \wedge \quad [m_2^{min}..m_2^{max}] \subseteq [e_2^{min}..e_2^{max}]$$

In diesem Fall lassen sich keine gültigen SoS-Konfigurationen aus dem SoS-Modell ableiten.

7.2.2 Automatisiert behebbare Inkonsistenzen in SoS-Modellen

Automatisiert behebbare Inkonsistenzen treten auf, wenn Beziehungen zwischen Kardinalitäten bestehen und eine Kardinalität mehr SoS-Konfigurationen zulässt als die andere. In diesen Fällen kann eindeutig bestimmt werden, welche die restriktivere Kardinalität ist. Daher kann diese Art von Inkonsistenz automatisiert behoben werden, indem die weniger restriktive Kardinalität angepasst wird.

Automatisiert behebbare Inkonsistenzen in SoS-Modellen können ebenfalls an den gleichen Stellen auftreten:

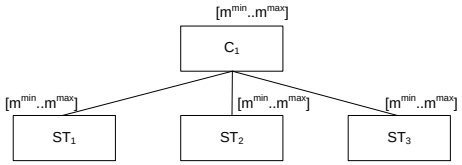
1. zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen
2. zwischen den Kardinalitäten von Constraintkanten und Kategorien bzw. Systemtypen

7.2.2.1 Automatisiert behebbare Inkonsistenzen zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen

Automatisiert behebbare Inkonsistenzen zwischen den Kardinalitäten von Oberkategorien und Unterkategorien bzw. Systemtypen treten auf, wenn die Kardinalität eines Knotens mehr SoS-Konfigurationen zulässt als verwandte Knoten.

Auftreten können die drei in Tabelle 7.10 illustrierten Fälle.

Tabelle 7.10: Automatisiert behebbare Inkonsistenzen in SoS-Modellen

Minimalkardinalität der Oberkategorie kleiner als Summe der Minimalkardinalitäten der Systemtypen $m^{min}(C_1) < \sum_{n=1}^{i} m^{min}(ST_i)$	
Maximalkardinalität der Oberkategorie größer als Summe der Maximalkardinalitäten der Systemtypen $m^{max}(C_1) > \sum_{n=1}^{i} m^{max}(ST_i)$	
Maximalkardinalität eines Systemtyps größer als die Maximalkardinalität der Oberkategorie abzügliche der Summe der Minimalkardinalitäten der anderen Systemtypen $m^{max}(ST_x) > m^{max}(C_1) - \sum_{n=1}^{i} m^{min}(ST_i) + m^{min}(ST_x)$	

Im ersten Fall ist die Minimalkardinalität m^{min} der Oberkategorie C_1 kleiner als die Summe aller Minimalkardinalitäten m^{min} der Unterkategorien bzw. Systemtypen ST. Dies suggeriert fälschlicherweise, dass SoS-Konfigurationen mit weniger Systemen der Kategorie C_1 zulässig sind als von den Unterkategorien bzw. Systemtypen gefordert. Diese Inkonsistenz kann behoben werden, indem die Minimalkardinalität der Oberkategorie gleich der Summe der Minimalkardinalitäten der Unterkategorien bzw. Systemtypen gesetzt wird. Das heißt:

$$m^{min}(C_1) = \sum_{n=1}^{i} m^{min}(ST_i)$$

Im zweiten Fall ist die Maximalkardinalität m^{max} der Oberkategorie C_1 größer als die Summe aller Maximalkardinalitäten m^{max} der Unterkategorien bzw. Systemtypen ST. Dies suggeriert, dass SoS-Konfigurationen mit mehr Systemen der Kategorie C_1 zulässig sind als von den Unterkategorien bzw. Systemtypen erlaubt. Diese Inkonsistenz kann behoben werden, indem die Maximalkardinalität der Oberkategorie gleich der Summe der Maximalkardinalitäten der Unterkategorien bzw. Systemtypen gesetzt wird. Das heißt:

$$m^{max}(C_1) = \sum_{n=1}^{i} m^{max}(ST_i)$$

Im dritten Fall ist die Maximalkardinalität einer Unterkategorie bzw. eines Systemtyps ST größer als die Maximalkardinalität der Oberkategorie. Dies suggeriert fälschlicherweise, dass mehr Systeme der Unterkategorie bzw. des Systemtyps ST_i zulässig sind als von den Oberkategorie erlaubt. Diese Inkonsistenz kann behoben werden, indem die Maximalkardinalität der Unterkategorie bzw. des Systemtyps ST_i gleich den Maximalkardinalitäten der Oberkategorie abzüglich der Summe der Minmialkardinalitäten der anderen Unterkategorien bzw. Systemtypen gesetzt wird.

Das heißt:

$$m^{max}(ST_x) > m^{max}(C_1) - \sum_{n=1}^{i} m^{min}(ST_i) + m^{min}(ST_x)$$

Bei den Maximalkardinalitäten können diese automatisiert behebbaren Inkonsistenzen auch in Zusammenhang mit einer Maximalkardinalität von unendlich auftreten. Schnabel et al. (2016); Weckesser et al. (2016) bezeichnen diese Art von Anomalie als falsche Unendlichkeit.

7.2.2.2 Automatisiert behebbare Inkonsistenzen zwischen den Kardinalitäten von Constraintkanten und Kategorien bzw. Systemtypen

Automatisiert behebbare Inkonsistenzen können ebenfalls in Zusammenhang mit Constraints auftreten. Um zu erkennen, ob eine automatisiert behebbare Inkonsistenz vorliegt, müssen die Art der Constraintkante, die Kardinalitäten der Constraintkante und die Kardinalität der betroffenen Knoten berücksichtigt werden.

Die verschiedenen Fälle von automatisiert behebbaren Inkonsistenzen zwischen den Kardinalitäten von Requires-Constraints und Kategorien bzw. Systemtypen werden in diesem Abschnitt wieder anhand des Beispiels aus Abbildung 7.6 illustriert.

Wenn die Minimalkardinalität m_0^{min} des Knotens ST_0, von dem das Constraint ausgeht, null ist, aber die Summe der Minimalkardinalitäten, auf die die Constraintkante zeigt, nicht größer als die Maximalkardinalität der Requires-Constraintkante ist oder die Summe der Maximalkardinalitäten, auf die die Constraintkante zeigt, nicht größer als die Minimalkardinalität der Requires-Constraintkante ist, liegt eine Inkonsistenz vor.

Das heißt, wenn

$$m_0^{min} > 0 \quad \wedge \quad (\sum_{n=1}^{i} m_i^{min} \leq r^{max} \quad \vee \quad \sum_{n=1}^{i} m_i^{max} \geq r^{min})$$

Diese Inkonsistenz lässt sich automatisiert beheben, indem die Kardinalität $[m_1^{min}..m_1^{max}]$ des Knotens ST_1, von dem die Constraintkante ausgeht, auf [0..0] geändert wird.

Außerdem liegt eine automatisiert behebbare Inkonsistenz vor, wenn die Minimalkardinalität m_1^{min} des Knotens ST_1 von dem die Requires-Constraintkante ausgeht, größer null ist und die Kardinalität $[m_2^{min}..m_2^{max}]$ des Knotens ST_2 keine Teilmenge der Kardinalität $[r^{min}..r^{max}]$ der Constraintkante ist, jedoch eine Überlappung vorliegt. Das heißt, wenn

$$(m_1^{min} > 0) \quad \wedge \quad ([m_2^{min}..m_2^{max}] \nsubseteq [e_2^{min}..e_2^{max}]) \quad \wedge \quad (([m_2^{min}..m_2^{max}] \cap [e_2^{min}..e_2^{max}]) \neq \emptyset)$$

Diese Inkonsistenz lässt sich automatisiert beheben, indem die Kardinalität $[m_2..m_2^{max}]$ des Knotens ST_2, auf den die Constraintkante zeigt, auf die Schnittmenge der beiden Kardinalitäten $[m_2^{min}..m_2^{max}] \cap [r^{min}..r^{max}]$ geändert wird.

Die verschiedenen Fälle von automatisiert behebbaren Inkonsistenzen zwischen den Kardinalitäten von Excludes-Constraints und Kategorien bzw. Systemtypen werden in diesem Abschnitt ebenfalls anhand des Beispiels aus Abbildung 7.7 illustriert.

Eine automatisiert behebbare Inkonsistenz liegt bei einer Excludes-Constraintkante vor, wenn die Kardinalität des einen Knotens eine Teilmenge der Kardinalität der Constraintkante ist und bei dem anderen Knoten die Kardinalität keine Teilmenge der Kardinalität der Constraintkante ist, jedoch eine Überlappung vorliegt. Das heißt, wenn

$$([m_1^{min}..m_1^{max}] \subseteq [e_1^{min}..e_1^{max}]) \wedge ([m_2^{min}..m_2^{max}] \nsubseteq [e_2^{min}..e_2^{max}]) \wedge$$
$$(([m_2^{min}..m_2^{max}] \cap [e_2^{min}..e_2^{max}]) \neq \emptyset)$$

oder wenn

$$([m_1^{min}..m_1^{max}] \nsubseteq [e_1^{min}..e_1^{max}]) \wedge (([m_1^{min}..m_1^{max}] \cap [e_1^{min}..e_1^{max}]) \neq \emptyset) \wedge$$
$$([m_2^{min}..m_2^{max}] \subseteq [e_2^{min}..e_2^{max}])$$

liegt eine Inkonsistenz vor. Diese Inkonsistenz lässt sich automatisiert beheben, indem die Kardinalität des Knotens, die keine Teilmenge der Kardinalität der Constraintkante ist, angepasst wird. Die korrigierte Kardinalität ergibt sich aus der Schnittmenge zwischen der Kardinalität des Knotens und der Kardinalität der Constraintkante. Dadurch wird die Spezifizierung von ungültigen SoS-Konfigurationen verhindert.

Bei der automatisierten Behebung von Inkonsistenzen muss beachtet werden, dass die Änderung von Kardinalitäten erneut zu Inkonsistenzen führen kann. Wenn diese nicht automatisiert behebbar sind, zeigt das SoS-Modell keine gültigen SoS-Konfigurationen.

7.2.3 Beispiele für Inkonsistenzen in SoS-Modellen

Abbildung 7.8 zeigt Beispiele für Inkonsistenzen in dem SoS-Modell der Transportroboterflotte. Automatisiert behebbare Inkonsistenzen sind blau und nicht automatisiert behebbare Inkonsistenzen sind orange hervorgehoben.

Die Gesamtanzahl der Roboter ist auf 20 beschränkt, jedoch ist die Anzahl der Roboter vom Typ H nicht eingeschränkt. Hierbei handelt es sich um eine automatisiert behebbare Inkonsistenz an einer Dekompositionskante. Die Inkonsistenz kann automatisiert behoben werden, indem die Maximalkardinalität für Roboter vom Typ H auf 20 gesetzt wird.

Die Gesamtanzahl der Ladestationen wird durch die Kardinalität der Kategorie *Ladestation* auf eins festgelegt. Jedoch fordern die Kardinalitäten der Systemtypen *Schnellladestation* und *Allgemeine Ladestation*, dass jeweils mindestens eine Ladestation von beiden Typen vorhanden ist. Für die Auswahl der SoS-Konfigurationen bedeutet dies, dass keine gültigen SoS-Konfigurationen gewählt wurden und somit keine gültige Zielsicht erzeugt werden kann. Liegt diese Inkonsistenz in einer erzeugten SoS-Konfigurationssicht vor, bedeutet dies, dass für die gewünschten Ziele keine gültige SoS-Konfigurationssicht existiert.

Die Requires-Constraintkante von *Roboter S* zu der *Warenabladestation* D_{40} stellt eine automatisiert behebbare Inkonsistenz dar. Die Constraintkante spezifiziert, dass es mindestens eine *Warenabladestation* des Typs D_{40} in der SoS-Konfiguration geben muss. Jedoch lässt die Kardinalität an dem D_{40}-Knoten auch SoS-Konfigurationen zu. Durch eine Änderung dieser Kardinalität zu [1..3] wird diese Inkonsistenz behoben. Nicht automatisiert beheben lässt sich die Inkonsistenz bei der Requires-Constraintkante von *Roboter für schwere Lasten* zu der *Warenaufnahmestation* P_{40}. Auch hier spezifiziert die Constraintkante, dass mindestens eine *Warenaufnahmestation* des Typs P_{40} in der SoS-Konfiguration vorhanden sein muss. Jedoch schließt die Kardinalität am P_{40}-Knoten *Warenaufnahmestationen* dieses Typs aus.

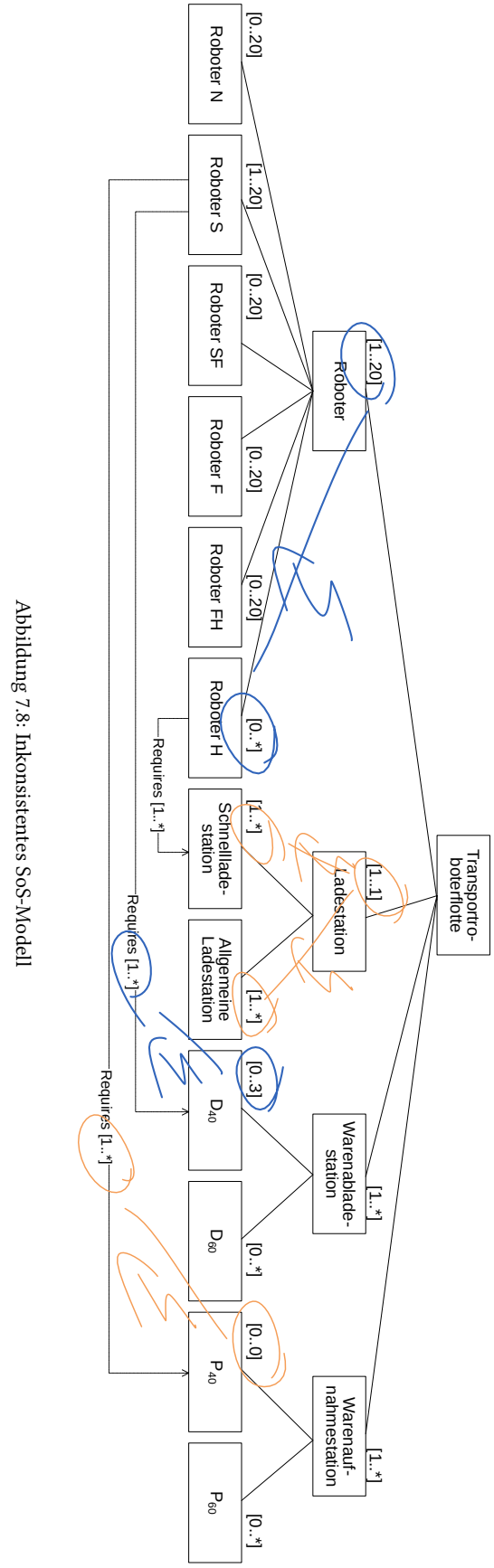


Abbildung 7.8: Inkonsistentes SoS-Modell

Teil III

Evaluation

8 Kapitel
Evaluationskonzept

Zur Evaluation des vorgeschlagenen Ansatzes wird überprüft, ob der Ansatz anwendbar ist und ob er einen Vorteil gegenüber dem aktuellen Stand der Wissenschaft bietet. Dies erfolgt anhand eines Fallbeispiels, einer prototypischen Implementierung und eines kontrollierten Experiments.

Abschnitt 8.1 stellt die Ziele vor, die mit der Evaluation des Ansatzes verfolgt werden. In Abschnitt 8.2 wird dargelegt, mit welchen Evaluationsmethoden die jeweiligen Evaluationsfragen beantwortet werden. Abschnitt 8.3 nennt abschließend die Beiträge, die die Evaluation leistet.

8.1 Ziele der Evaluation

Diese Arbeit leistet einen Beitrag zur Spezifikation und Validierung von Zielen von Systems-of-Systems. In diesem Kontext wurde ein Ansatz vorgestellt, der die Spezifikation von Zielen für Systems-of-Systems unterstützt und dem Nutzer die Möglichkeit bietet, automatisiert Sichten zu erstellen. Die vorgeschlagene Spezifikation der Ziele eines System-of-Systems in einer SoS-Zielspezifikation erlaubt eine präzise und kompakte Darstellung. Die automatisierte Generierung von Sichten unterstützt die Validierung von SoS-Zielspezifikationen durch eine für die Validierung optimierte Darstellung. In diesem Teil der Arbeit wird dieser Beitrag evaluiert. Zur Bewertung, ob der vorgeschlagene Ansatz die in Kapitel 3 definierte Forschungslücke schließt, werden im Rahmen der Evaluation die folgenden zwei Evaluationsfragen beantwortet.

1. E1: Ist der Ansatz anwendbar?

2. E2: Erweist sich der Ansatz als vorteilhaft gegenüber dem aktuellen Stand der Wissenschaft?

8.2 Evaluationsmethoden

Zur Evaluation der Anwendbarkeit und der Vorteilhaftigkeit des vorgeschlagenen Ansatzes werden unterschiedliche Evaluationsmethoden eingesetzt.

8.2.1 Anwendbarkeit des Ansatzes

Durch die Evaluation der Anwendbarkeit soll gezeigt werden, dass der vorgestellte Ansatz in der Praxis prinzipiell anwendbar ist. Die Bewertung der Anwendbarkeit des Ansatzes erfolgt anhand von zwei Evaluationsmethoden. Die Anwendbarkeit der SoS-Zielspezifikation wird an einem Fallbeispiel gezeigt. Das Fallbeispiel verdeutlicht, dass mithilfe der SoS-Zielspezifikation die Ziele eines System-of-Systems dokumentiert werden können. Die Anwendbarkeit der automatisierten Sichtengenerierung wird durch eine prototypische Implementierung erwiesen. Die prototypische Implementierung legt nahe, dass aus der SoS-Zielspezifikation automatisiert Sichten generiert werden können. Der Prototyp wurde sowohl auf seine Funktion als auch auf seine Performanz getestet. Dadurch wird sichergestellt, dass sich die Sichten innerhalb eines akzeptablen Zeitraums erstellen lassen.

8.2.2 Vorteilhaftigkeit des Ansatzes

Durch die Evaluation der Vorteilhaftigkeit soll belegt werden, dass der Ansatz einen wertvollen Beitrag zum Stand der Wissenschaft leistet. Da die Spezifikation von Anforderungen für variable Systeme in einem Basis- und einem Variabilitätsmodell als etabliert angesehen wird und der Nutzen vielfach erwiesen wurde (vgl. Kapitel 3.3), konzentriert sich die Evaluation der Vorteilhaftigkeit auf die automatisiert generierten Sichten. In einem kontrollierten Experiment wird die Validierung der SoS-Zielspezifkation mit der Validierung der Sichten verglichen. Dies erlaubt die quantitative Bewertung des Nutzens der Sichtenerstellung zur Validierung von SoS-Zielspezifikationen. Im Rahmen des Experiments wird nicht der Prozess der Sichtenerstellung evaluiert, d. h. die Auswahl der gewünschten Ziele bzw. SoS-Konfigurationen. Diese wurden im Rahmen des Experiments vorgegeben.

8.3 Evaluationbeiträge

Diese Arbeit leistet die folgenden Beiträge zur Evaluation:

- In Abschnitt 9.1 zeigt ein Fallbeispiel, dass der vorgestellte Ansatz prinzipiell anwendbar ist.
- In Abschnitt 9.2 wird eine prototypische Implementierung vorgestellt, die verdeutlicht, dass die Sichtenerstellung automatisierbar ist.
- In Kapitel 10 leitet sich aus einem kontrollierten Experiment ab, welche Vorteile die Nutzung des Ansatzes bietet.

9 Kapitel
Evaluation der Anwendbarkeit

Die Anwendung des vorgeschlagenen Ansatzes an einem Fallbeispiel aus der Robotik-Industrie zeigt, dass der Ansatz prinzipiell anwendbar ist. Durch die prototypische Implementierung des Ansatzes wird gezeigt, dass die Sichten automatisiert aus der SoS-Zielspezifikation erstellt werden können und dies in akzeptabler Laufzeit geschieht.

Dieses Kapitel stellt die Evaluation der Anwendbarkeit vor. In Abschnitt 9.1 wird die Anwendung des Ansatzes an einem realitätsnahen Fallbeispiel gezeigt. Abschnitt 9.2 greift eine prototypische Implementierung auf, die die technische Umsetzbarkeit des Ansatzes belegt.

9.1 Fallbeispiel

Dieser Abschnitt stellt das Fallbeispiel, das zur Evaluation der Anwendbarkeit genutzt wurde, sowie die Ergebnisse der Anwendung vor. Der Aufbau orientiert sich an den Empfehlungen für Berichte von Fallstudien von Runeson et al. (2012).

9.1.1 Ziel des Fallbeispiels

Anhand des Fallbeispiels soll untersucht werden, ob der vorgeschlagene Ansatz prinzipiell anwendbar ist, das heißt, ob für das Fallbeispiel eine SoS-Zielspezifikation erstellt werden kann und hieraus Zielsichten und SoS-Konfigurationssichten abgeleitet werden können.

9.1.2 Beschreibung des Fallbeispiels

Als Fallbeispiel wurde eine Transportorboterflotte gewählt. Das Fallbeispiel entstand in enger Zusammenarbeit mit einem Unternehmen aus der Robotik-Industrie.

Die Transportroboterflotte besteht aus autonomen Transportrobotern, Ladestationen zum Aufladen der Transportroboterakkus, Annahmestationen zur Übergabe von Transportgütern an einen Transportroboter sowie Abladestationen zum Abladen von Transportgütern von einem Transportroboter. Die Transportroboterflotte soll zwischen einem und zwanzig Transportroboter sowie zwischen einer und zehn Ladestationen besitzen. Es soll jeweils mindestens eine und maximal fünf Annahme- und Abladestationen in der Transportroboterflotte geben.

Insgesamt kann die Transportroboterflotte acht verschiedene Typen von Transportrobotern beinhalten. Außerdem können drei verschiedene Typen von Ladestationen Teil der Transportroboterflotte sein. Jeweils vier verschiedene Typen von Annahme- und Abladestationen sind innerhalb der Transportroboterflotte möglich. Die genaue Anzahl der Transportroboter, Ladestationen, Annahme- und Abladestationen sind nicht weiter eingeschränkt. Das bedeutet, die Transportroboterflotte kann von jedem Transportrobotertyp maximal 20 Transportroboter besitzen, von jedem Ladestationstyp maximal zehn und von jedem Annahme- und Abladestationstyp maximal fünf (jeweils, so lange die Maximalanzahl nicht überschritten ist).

Die Transportroboterflotte hat das Ziel, Waren zu transportieren. Dabei muss sie Transportaufträge verteilen, Fahrstrecken berechnen, Kollisionen vermeiden, die Transporte durchführen und die Akkus verwalten. Die Verteilung der Transportaufträge erfolgt über ein Auktionsverfahren, bei dem die Transportroboter Gebote abgeben. Die Gebote, die ein Transportroboter abgibt, sind abhängig von seinem Akkustand, seinen Fähigkeiten und den Eigenschaften der Annahme- und Abladestation. Die Verteilung der Transportaufträge kann auf Basis einer Optimierung für individuelle Transportroboter oder für die ganze Transportroboterflotte durchgeführt werden. Durch die Optimierung der Vergabe sollen Wartezeiten an Ladestationen sowie Stillstände verhindert werden.

Kollisionen werden durch die Erkennung von Hindernissen und falls nötig Notbremsungen vermieden. Hindernisse erkennen Transportroboter über ihre Sensoren. Informationen über erkannte Hindernisse werden von dem Transportroboter, der dieses Hindernisse erkannt hat, an die anderen Transportroboter weitergegeben.

Zur Berechnung der Fahrstrecken wird von den Transportrobotern eine Karte erstellt und aktuell gehalten. Die Fahrstrecken müssen jeweils die Fahrstrecken anderer Transportroboter berücksichtigen. Für den Transport müssen die Transportroboter die Ware an der Annahmestation annehmen, zur Abladestation transportieren und dort abladen. Die Ware kann besonders schwer, besonders groß oder flüssig sein. Die Akkus der Transportroboter können neben einer normalen Aufladung auch besonders schnell oder besonders schonend geladen werden.

Um Stillstand zu vermeiden, muss die Transportroboterflotte mindestens sechs Transportroboter und drei Ladestationen aufweisen. Damit mehrere Transportaufträge gleichzeitig bearbeitet und Waren zwischen Transportrobotern übergeben werden können, muss die Transportroboterflotte mindestens zwei Transportroboter besitzen. Nicht alle Transportrobotertypen erlauben die Annahme und das Abladen von Waren auf unterschiedlichen Höhen. Dafür wird mindestens ein Transportroboter des Typs III, V, VI oder VII benötigt. Um Wartezeiten an Ladestationen zu vermeiden, muss es mindestens fünf Ladestationen geben. Transportroboter der Typen I, II, IV, V und VII sind besonders wartungsaufwendig.

Die schnelle Aufladung der Akkus erfordert mindestens eine Ladestation des Typs II. Zur schonenden Aufladung der Akkus wird mindestens eine Ladestation des Typs III benötigt. Transportroboter des Typs II und VII können Ware besonders schnell transportieren. Transportroboter des Typs V und VI können besonders schwere Ware transportieren. Transportroboter

des Typs I, II und V sind in der Lage, Flüssigkeiten zu transportieren. Besonders große Ware fallen in den Zuständigkeitsbereich von Transportrobotern des Typs VII.

9.1.3 Modellierung des Fallbeispiels

Dieser Abschnitt stellt die Ergebnisse der Anwendung des Ansatzes auf das Fallbeispiel vor. Gezeigt werden die Zielspezifikation, das SoS-Modell und die X-Links im Rahmen der SoS-Zielspezifikation. Ebenfalls werde jeweils ein Beispiel für die Zielsicht und ein Beispiel für die SoS-Konfigurationssicht aufgegriffen.

Zielspezifikation

Das GRL-Zielmodell für die Transportroboterflotte ist in Abbildung 9.1 dargestellt. Es zeigt die Ziele der Transportroboterflotte (vgl. 9.1.2). Das Oberziel ist der Transport von Waren. Die weiteren Ziele der Transportroboterflotte wurden strukturiert und miteinander in Beziehung gesetzt. Neben der Dekompositonshierarchie sind auch diverse Contributions spezifiziert. Bspw. hat die Task Schnelles Aufladen einen positiven Einfluss auf die Verhinderung eines Stillstands. Ebenfalls spezifiziert sind Ressourcen, die benötigt werden, wie bspw. Daten der Transportroboter und Annahme- und Abladestationen, die bei der Vergabe von Transportaufträgen berücksichtigt werden müssen.

SoS-Modell

Das SoS-Modell für die Transportroboterflotte ist in Abbildung 9.2 dargestellt. Es zeigt die möglichen in Abschnitt 9.1.2 beschriebenen gültigen SoS-Konfigurationen der Transportroboterflotte. Das SoS-Modell spezifiziert, dass die Transportroboterflotte zwischen einem und 20 Transportrobotern beinhalten soll und dass es acht verschiedene Typen von Transportrobotern in der Flotte geben kann. Außerdem ist spezifiziert, dass es insgesamt zwischen einer und zehn Ladestationen geben muss, die jeweils einem von drei verschiedenen Typen angehören müssen. Die zulässige Anzahl von Annahmestationen und Abladestationen sowie die verschiedenen Arten sind ebenfalls spezifiziert.

SoS-Zielspezifikation

Das SoS-Zielmodell für die Transportroboterflotte ist in Abbildung 9.3 dargestellt. Es zeigt die Zusammenhänge zwischen den SoS-Konfigurationen und dem Zielmodell (vgl. 9.1.2). Die Abbildung vermittelt einen Eindruck von der Komplexität. Die verschiedenen Farben der X-Links dienen dazu, die Unterscheidung zu vereinfachen, und besitzen darüber hinaus keine weitere Bedeutung.

Abbildung 9.1: Zielmodell des Fallbeispiels

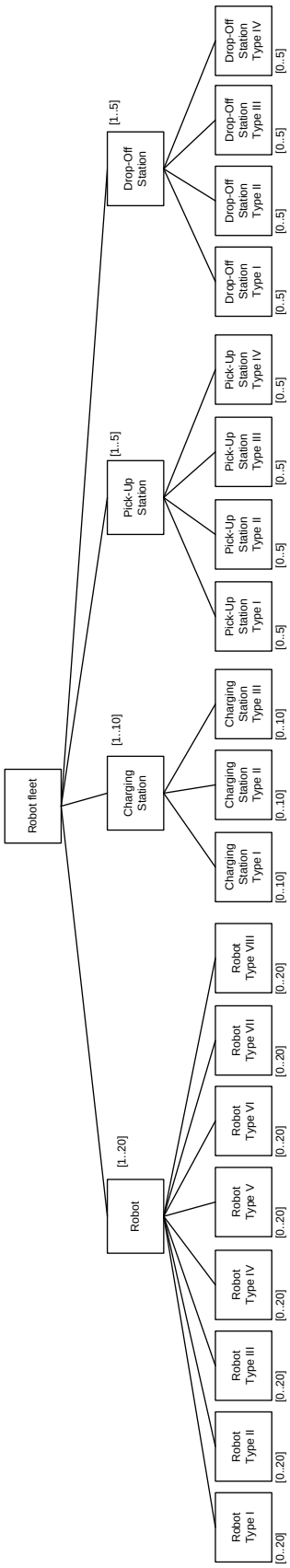


Abbildung 9.2: SoS-Modell des Fallbeispiels

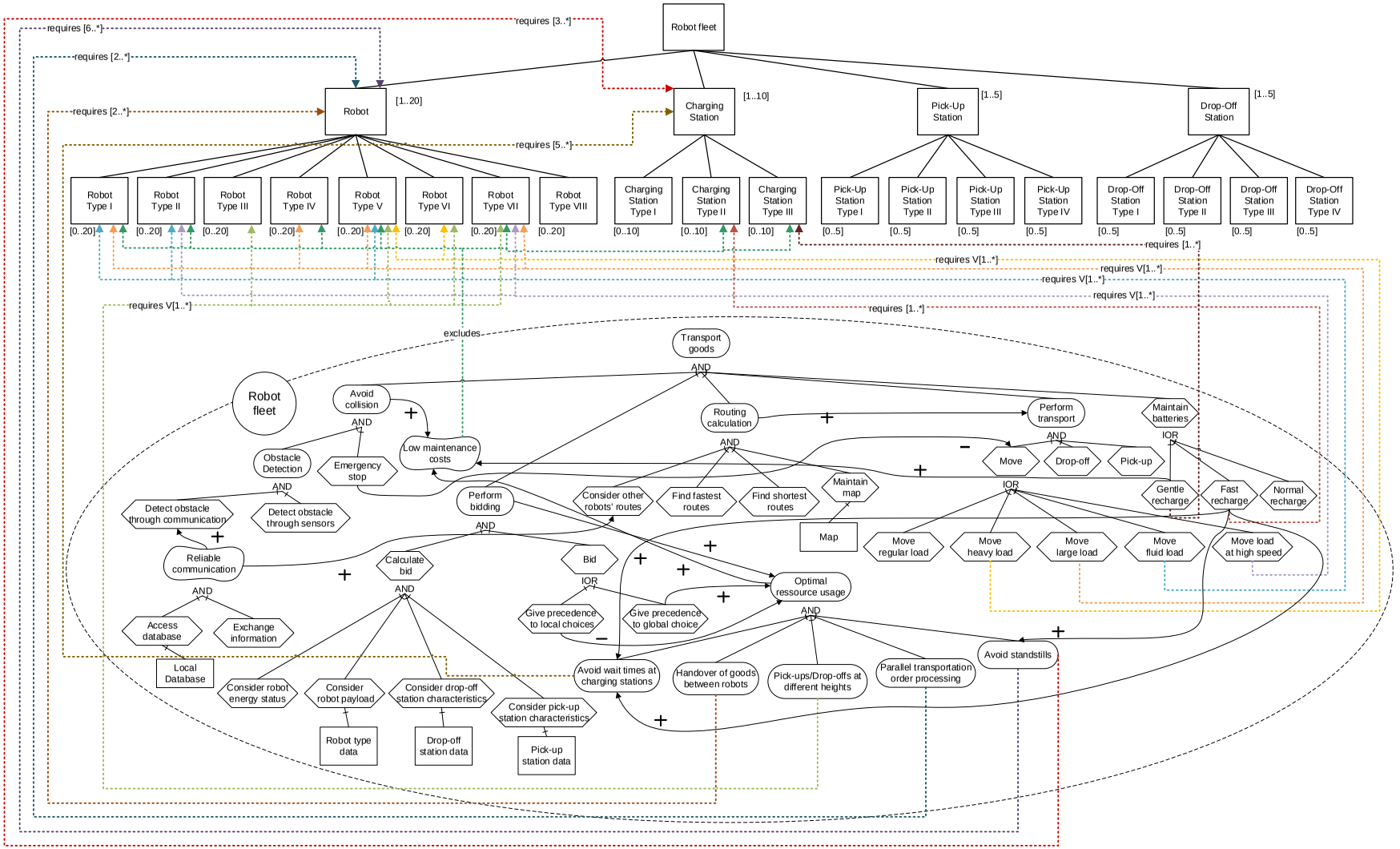


Abbildung 9.3: SoS-Zielspezifikation des Fallbeispiels

Beispielhafte Sichterstellung: Zielsicht

Zur Erstellung einer beispielhaften Zielsicht wurden die SoS-Konfigurationen mit den folgenden Eigenschaften ausgewählt:

- 1-2 Charging Stations Type I
- 1-2 Charging Station Type III
- 1-3 Robots Type II
- 1-3 Robots Type V
- 1 Robot Type VIII
- 2-5 Pick-Up Stations Type II
- 2-5 Drop-Off Stations Type II

Abbildung 9.4 zeigt die generierte Zielsicht. Die Ziele *Wartezeiten an Ladestationen vermeiden (Avoid wait times at charging stations)* und *Schnelles Laden (Fast recharge)* können von den ausgewählten SoS-Konfigurationen nicht erfüllt werden, da die ausgewählten SoS-Konfigurationen weniger als fünf Ladestationen und keine Ladestation vom Typ II enthalten. Das Ziel *Stillstand vermeiden (Avoid standstills)* ist nur für einen Teil der ausgewählten SoS-Konfigurationen erfüllbar. Die Erfüllung dieses Ziels erfordert mindestens drei Ladestationen und sechs Roboter. Die Auswahl schließt jedoch auch Transportroboterflotten mit nur drei Robotern und zwei Ladestationen ein. Die Ziele bzw. Tasks *Übergabe von Waren zwischen Robotern (Handover of goods between robots)*, *Annahmen und Abgaben auf unterschiedlichen Höhen (Pick-ups/Drop-offs at different heights)*, *Parallele Verarbeitung von Transportaufträgen (Parallel transportation order processing)*, *Schwere Ladung bewegen (Move heavy load)*, *Große Ladung bewegen (Move large load)*, *Flüssige Ladung bewegen (Move fluid load*) und *Ladung schnell bewegen (Move load at high speed)* können für die ausgewählten SoS-Konfigurationen erfüllt werden. Gemäß den Regeln der Erfüllbarkeitsprüfung können ebenfalls die Tasks *Bewegen (Move)* und *Batterien pflegen (Maintain batteries)* erfüllt werden. Das Ziel *Optimale Ressourcennutzung (Optimal resource ussage)* ist jedoch nicht erfüllbar, da das AND-dekomponierte Unterziel *Wartezeiten an Ladestationen vermeiden (Avoid wait times at charging stations)* nicht erfüllt werden kann. Die Erfüllbarkeit aller weiteren intentionalen Elemente ist unabhängig von der Auswahl der SoS-Konfiguration.

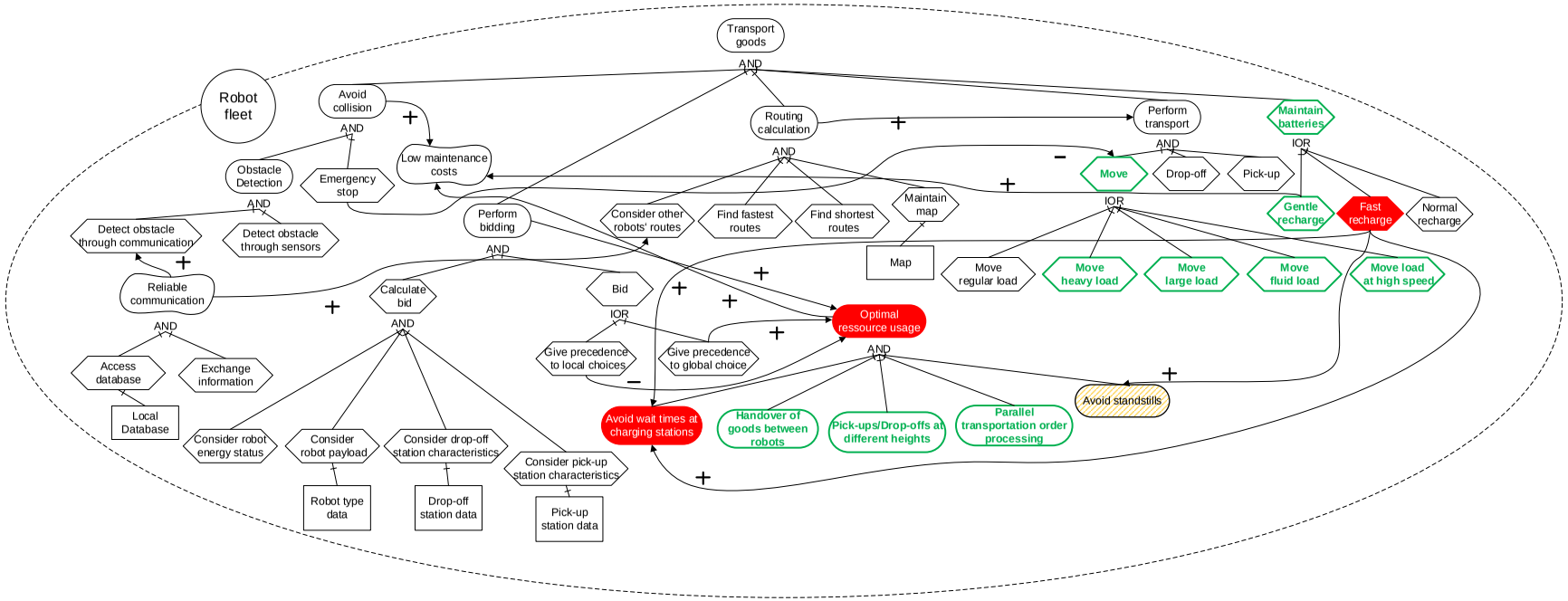


Abbildung 9.4: Beispielhafte Zielsicht des Fallbeispiels

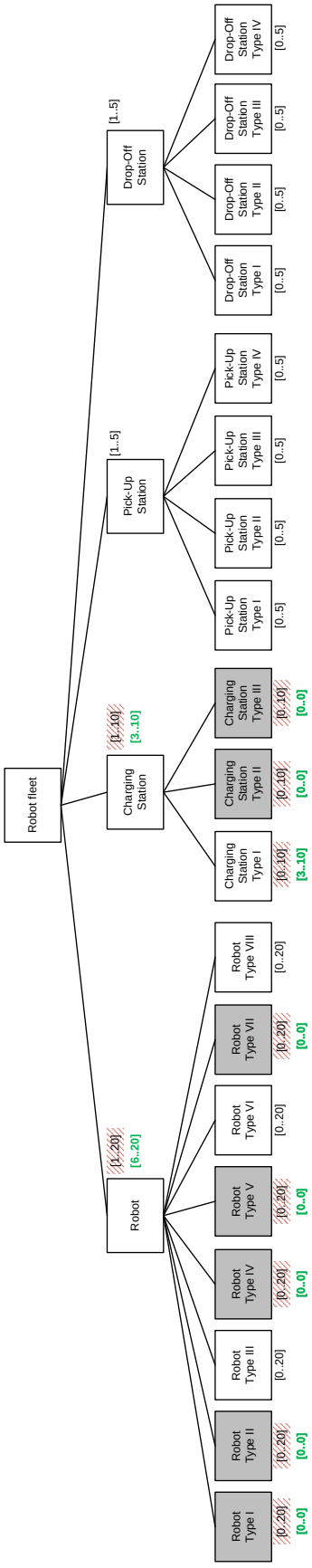


Abbildung 9.5: Beispielhafte SoS-Konfigurationssicht des Fallbeispiels

Beispielhafte Sichterstellung: SoS-Konfigurationssicht

Zur Erstellung einer beispielhaften SoS-Konfigurationssicht wurden die folgenden Ziele als gewünscht ausgewählt:

- Stillstand vermeiden (Avoid standstills)
- Niedrige Wartungskosten (Low maintenance costs)

Es wurden keine Ziele als ungewünscht ausgewählt.

Abbildung 9.5 zeigt die generierte SoS-Konfigurationssicht. Gemäß der Sicht muss die Transportroboterflotte sechs bis 20 Roboter enthalten. Allerdings keine Roboter des Typs I, II, IV und V. Außerdem muss die Transportroboterflotte drei bis zehn Ladestationen besitzen. Diese müssen alle vom Typ I sein, da die anderen beiden Ladestationstypen nicht Teil der Transportroboterflotte sein dürfen. Es müssen ebenso jeweils mindestens eine und maximal fünf Annahmestationen und Abladestationen zu der Transportroboterflotte gehören.

9.2 Prototypische Werkzeugunterstützung

Um die technische Umsetzbarkeit des vorgeschlagenen Ansatzes zu evaluieren, wurde dieser prototypisch implementiert. Dieses Kapitel bietet einen Überblick über den Aufbau und die Funktionsweise des Prototyps. Außerdem wird anhand von Laufzeitmessungen gezeigt, dass Sichten für realistisch große Spezifikation in annehmbarer Zeit generiert werden können.

9.2.1 Aufbau und Funktionsweise des Prototyps

Der Prototyp gliedert sich in zwei Teile. Für die Erstellung der SoS-Zielspezifikation bestehend aus Zielmodell, SoS-Modell und X-Links sowie für die Anzeige der Sichten wird das kommerzielle Modellierungstool Microsoft Visio verwendet. Microsoft Visio erlaubt die Anpassung an bisher nicht unterstützte Modellierungssprachen durch die Definition von Schablonen. Visio-Schablonen legen die verfügbaren Modellelemente fest. Abbildung 9.6 zeigt die definierten Schablonen zur Spezifikation von Zielmodell, SoS-Modell und X-Links.

Für die Sichtenerstellung und Konsistenzprüfung wurde die Windows-Forms-Applikation *CPS View Generator* in der Programmiersprache C# entwickelt. Windows-Forms-Applikationen können wie Microsoft Visio auf Windows-Rechnern ausgeführt werden. Der *CPS View Generator* erlaubt den Import von mit Microsoft Visio erstellten VSDX-Dateien. Dabei ist darauf zu achten, dass für die Erstellung der Spezifikation die vorgegebene Schablone verwendet werden muss, um sicherzustellen, dass der *CPS View Generator* alle Modellelemente richtig identifiziert. Beim Einlesen der VSDX-Datei erfolgt eine syntaktische Prüfung, ob es sich um eine mit der vorgegebenen Schablone erstellte Spezifikation von Zielmodell, SoS-Modell und X-Links handelt. Falls andere Modellelemente vorhanden sind oder notwendige Modellelemente fehlen, wird dem Nutzer eine Fehlermeldung angezeigt.

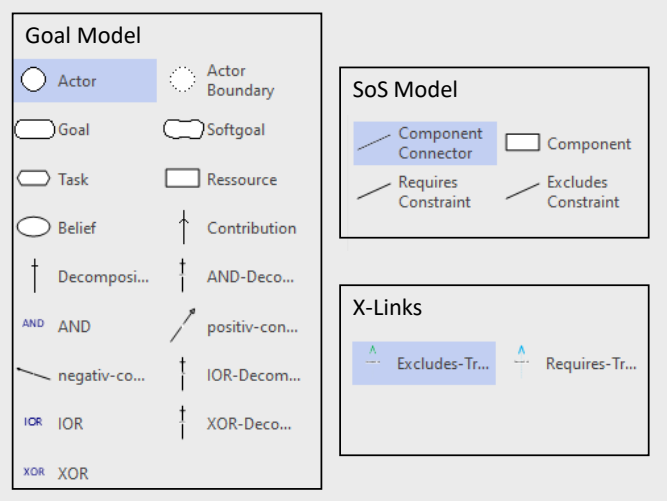


Abbildung 9.6: Visio-Schablonen zur Erstellung der Spezifikation

Nach dem Import der Spezifikation hat der Nutzer die Wahl, ob er eine Ziel- oder eine SoS-Konfigurationssicht erstellen möchte. Zur Erstellung einer Zielsicht werden dem Nutzer alle Kategorien und Systemtypen angezeigt und er kann für diese die Kardinalitäten festlegen. Die Benutzeransicht ist in Abbildung 9.7 oben dargestellt. Um Fehler zu verhindern, erlaubt der *CPS View Generator* nur die Definition von Kardinalitäten innerhalb der in der Spezifikation festgelegten Kardinalitäten. Wenn der Nutzer die Auswahl beendet hat, wird diese auf Inkonsistenzen überprüft. Bei einer inkonsistenten Auswahl erhält der Nutzer eine Fehlermeldung und die Möglichkeit, seine Auswahl zu ändern. Basierend auf der Auswahl wird die Zielsicht generiert. Die Zielsicht wird ebenfalls auf Inkonsistenzen geprüft und der Nutzer wird auf in der Zielsicht vorhandene Inkonsistenzen aufmerksam gemacht. Der *CPS View Generator* zeigt dem Nutzer die Namen der in der Zielsicht nicht erfüllbaren intentionalen Elemente an, wie in Abbildung 9.7 unten dargestellt ist, und bietet die Möglichkeit, diese Sicht als VSDX-Datei zu exportieren. Die erstellte VSDX-Datei kann nun mit Microsoft Visio geöffnet werden und zeigt dort die erstellte Zielsicht an.

Die Erstellung einer SoS-Konfigurationssicht erfolgt analog. Dem Nutzer werden alle intentionalen Elemente des Zielmodells angezeigt und er kann die gewünschten Ziele auswählen. Die Benutzeransicht ist in Abbildung 9.8 oben dargestellt. Auch hier findet nach der Auswahl eine Prüfung auf Konsistenz statt. Basierend auf der Auswahl wird die SoS-Konfigurationssicht generiert. Diese wird ebenfalls auf Inkonsistenzen überprüft und der Nutzer wird informiert, wenn Inkonsistenzen vorliegen. Der *CPS View Generator* zeigt, wie in Abbildung 9.8 unten dargestellt ist, die ausgeschlossenen Kategorien und Systemtypen sowie die veränderten Kardinalitäten der Kategorien und Systemtypen für die erstellte SoS-Konfigurationssicht an und ermöglicht es dem Nutzer, die SoS-Konfigurationssicht als VSDX-Datei zu exportieren. Mithilfe

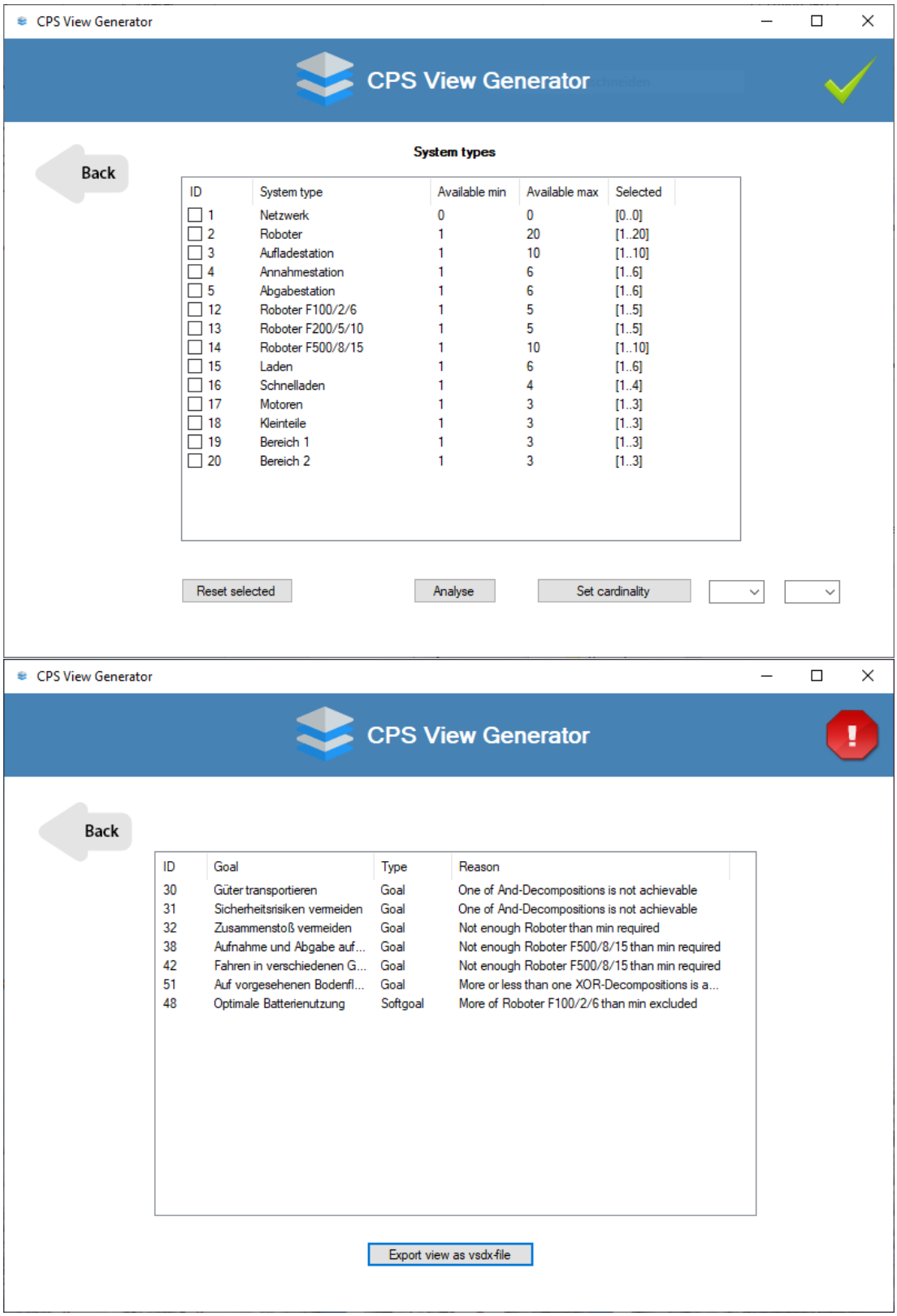


Abbildung 9.7: Erstellung einer Zielsicht

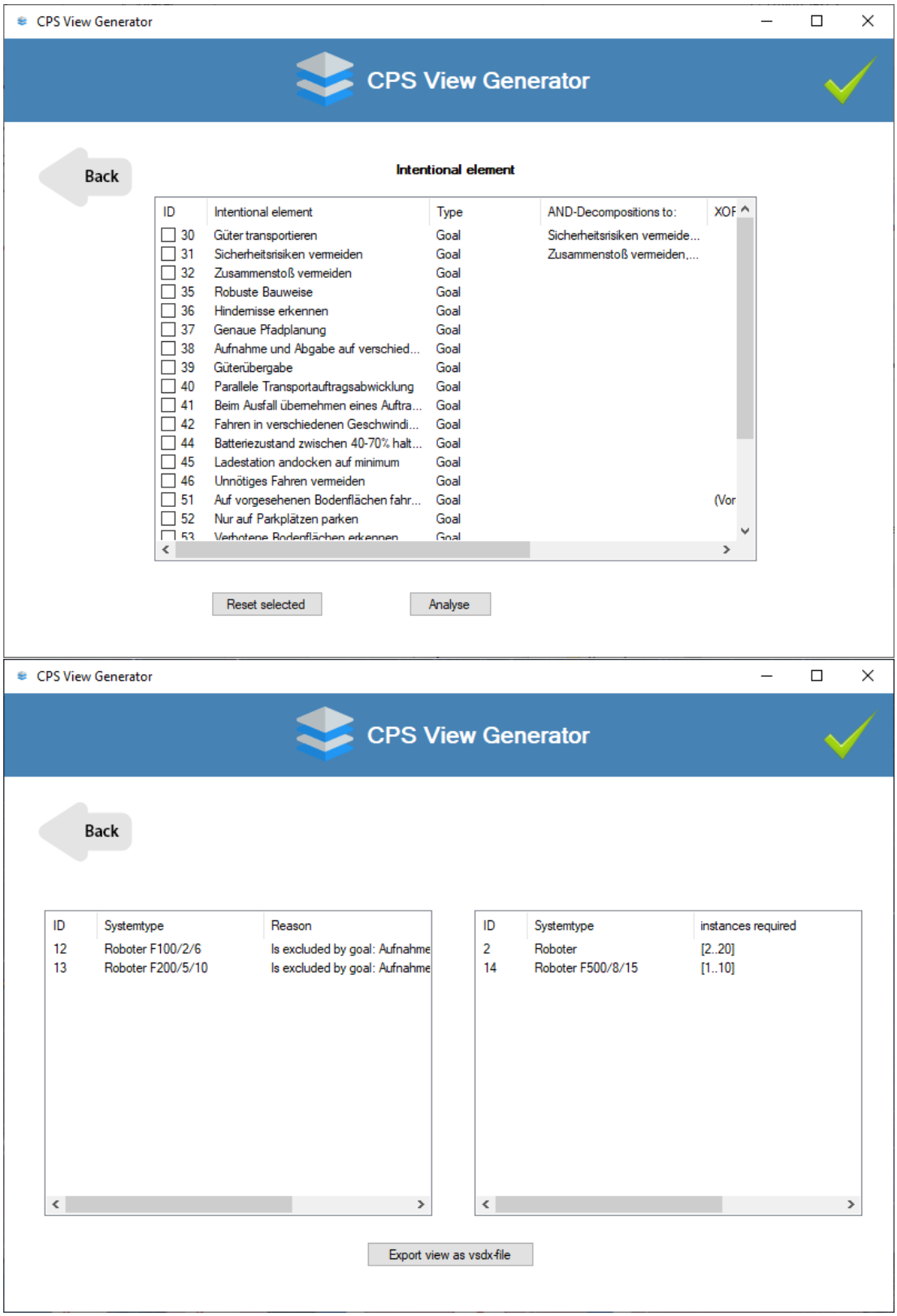


Abbildung 9.8: Erstellung einer SoS-Konfigurationssicht

von Microsoft Visio kann die Sicht betrachtet werden. Abbildung 9.8 illustriert die Erstellung einer SoS-Konfigurationssicht mit dem *CPS View Generator*.

9.2.2 Laufzeitmessungen

Die Laufzeitmessungen wurden mit dem Leistungs-Profiler von Microsoft Visual Studio auf einem Windows PC mit Intel Core i7-8550U @ 1,8 GHz, 1992 MHz, 4 Kerne Prozessor und 16GB Arbeitsspeicher durchgeführt. Um zu ermitteln, welche Auswirkungen unterschiedliche Größenspezifikationen auf die Laufzeit der Sichtenerstellung und der Konsistenzprüfung haben, wurden verschieden große Testspezifikationen erstellt, bei denen die Anzahl der intentionalen Elemente im Zielmodell, der Systemtypelemente im SoS-Modell und der X-Links variiert [7]

Es wurden Testspezifikationen mit 1, 10, 20, 50, 100, 200, 500 und 1000 X-Links erstellt. Alle Testspezifikationen enthielten außerdem 1001 intentionale Elemente (ein Wurzelelement und 1000 Unterelemente), 1000 Systemtypelemente und ein Wurzelelement für das SoS-Modell.

Laufzeit bei einer unterschiedlichen Anzahl von X-Links

Abbildung 9.9 zeigt die Laufzeit zur Erstellung von Zielsichten und SoS-Konfigurationssichten einschließlich aller Konsistenzprüfungen. Es wurden immer alle intentionalen Elemente bzw. alle Systemtypen ausgewählt. Die Testspezifikationen wurden so erstellt, dass keine Inkonsistenzen in der Auswahl und in den Sichten auftreten, um sicherzustellen, dass eine vollständige Konsistenzprüfung erfolgt.

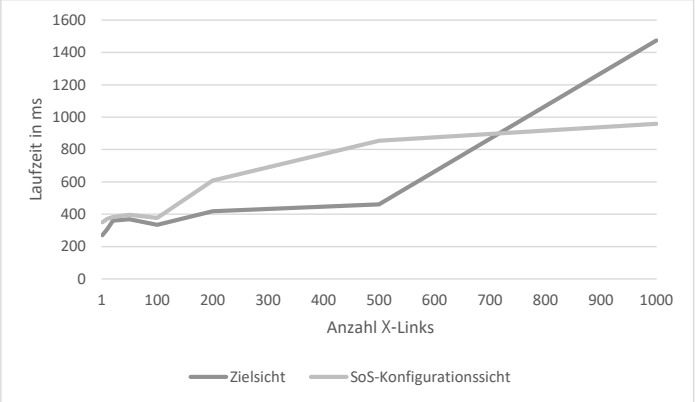


Abbildung 9.9: Einfluss der Anzahl der X-Links auf die Dauer der Sichterstellung

[7] Die verwendeten Testspezifikationen sind vom Umfang und teilweise vom Aufbau nicht realistisch. Sie wurden so erstellt, um a) zu zeigen, dass Spezifikationen mit einem Umfang, der größer als realistisch zu erwarten ist, in einem akzeptablen Zeitraum verarbeitet werden können, und um b) zu zeigen, welche Modellelemente einen besonders großen Einfluss auf die Laufzeit haben.

Wie zu sehen ist, dauert eine Sichtenerstellung bei einem bis 100 X-Links (und jeweils 1000 intentionalen Unterelementen und 1000 Systemtypelementen) zwischen 300 und 400 Millisekunden. Bei 200 X-Links dauert die Erstellung einer Zielsicht ungefähr 400 Millisekunden, diejenige einer SoS-Konfigurationssicht rund 600 Millisekunden. Die Erstellung einer Zielsicht bei 500 X-Links erfolgt in knapp 500 Millisekunden, eine SoS-Konfigurationssicht benötigt knapp 900 Millisekunden. Bei 1000 X-Links steigt die Dauer auf knapp 1500 Millisekunden für die Erstellung der SoS-Konfigurationssicht und auf knapp 1000 Millisekunden für die Erstellung der Zielsicht.

Laufzeit bei einer unterschiedlichen Anzahl von intentionalen Elementen im Zielmodell

Es wurden außerdem Testspezifikationen mit 1, 10, 20, 50, 100, 200, 500 und 1000 intentionalen Unterelementen erstellt. Alle Testspezifikationen enthielten des Weiteren 1000 X-Links, ein Wurzelelement für das Zielmodell, 1000 Systemtypelemente und ein Wurzelelement für das SoS-Modell. Abbildung 9.10 zeigt die Laufzeit zur Erstellung von Zielsichten und SoS-Konfigurationssichten einschließlich aller Konsistenzprüfungen. Zum besseren Vergleich ist auch noch einmal die Laufzeit für die Testspezifikation mit 1000 intentionalen Unterelementen, 1000 Systemtypelementen und 1000 X-Links dargestellt. Es wurden immer alle intentionalen Elemente bzw. alle Systemtypen ausgewählt. Die Testspezifikationen wurden so erstellt, dass keine Inkonsistenzen in der Auswahl und in den Sichten auftreten, um sicherzustellen, dass eine vollständige Konsistenzprüfung erfolgt.

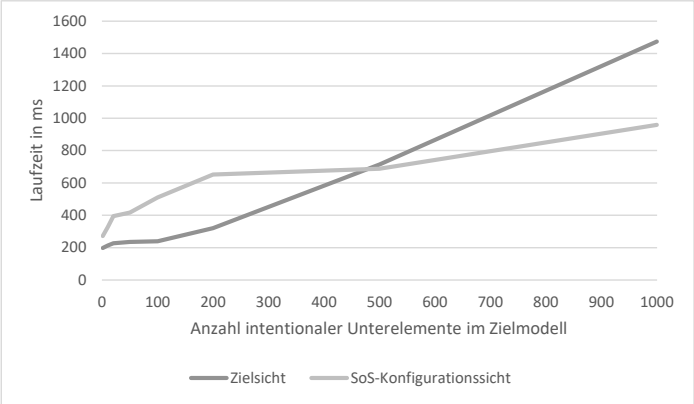


Abbildung 9.10: Einfluss der Anzahl der intentionalen Elemente im Zielmodell auf die Dauer der Sichterstellung

Die Sichtenerstellung dauert bei Spezifikationen mit bis zu 100 Unterelementen im Zielmodell nicht länger als 600 Millisekunden für SoS-Konfigurationssichten und 300 Millisekunden für Zielsichten. Bei 200 intentionalen Unterelementen dauert die Erstellung einer Zielsicht knapp 400 Millisekunden, diejenige einer SoS-Konfigurationssicht knapp 700 Millisekunden.

Innerhalb von ungefähr 700 Millisekunden können Zielsichten und SoS-Konfigurationssichten mit 500 intentionalen Unterlementen in der SoS-Zielspezifikation erstellt werden.

Laufzeit bei einer unterschiedlichen Anzahl von Systemtypen im SoS-Modell

Außerdem wurden Testspezifikationen mit 1, 10, 20, 50, 100, 200, 500 und 1000 Systemtypelementen erstellt. Alle Testspezifikationen enthielten des Weiteren 1000 X-Links, 1001 intentionale Elemente (ein Wurzelelement und 1000 Unterelemente) und ein Wurzelelement für das SoS-Modell. Abbildung 9.11 zeigt die Laufzeit für die Erstellung von Zielsichten und SoS-Konfigurationssichten einschließlich aller Konsistenzprüfungen. Zum besseren Vergleich ist auch noch einmal die Laufzeit für die Testspezifikation mit 1000 intentionalen Unterelementen, 1000 Systemtypelementen und 1000 X-Links dargestellt. Es wurden immer alle intentionalen Elemente bzw. alle Systemtypen ausgewählt. Die Testspezifikationen wurden so erstellt, dass keine Inkonsistenzen in der Auswahl und in den Sichten auftreten, um sicherzustellen, dass eine vollständige Konsistenzprüfung erfolgt.

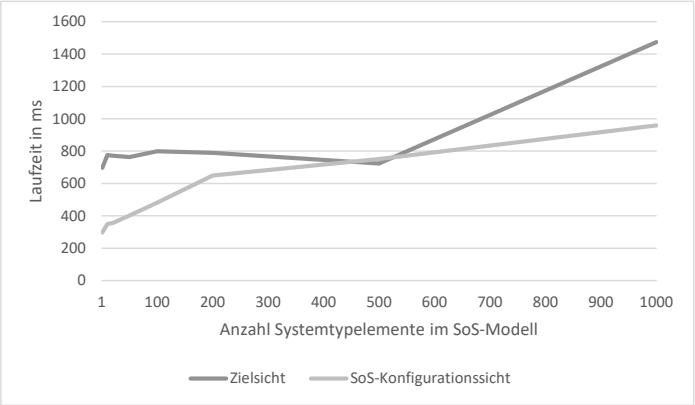


Abbildung 9.11: Einfluss der Anzahl der Systemtypen im SoS-Modell auf die Dauer der Sichterstellung

Die Sichtenerstellung dauert bei Spezifikationen mit bis zu 100 Systemtypelementen im SoS-Modell nicht länger als 800 Millisekunden für Zielsichten und 500 Millisekunden für SoS-Konfigurationssichten. Bei 200 Systemtypelementen dauert es ungefähr 800 Millisekunden, eine Zielsicht zu erstellen, und ungefähr 700 Millisekunden, eine SoS-Konfigurationssicht zu erstellen. Bei 500 Systemtypelementen wurden ähnliche Laufzeitwerte gemessen wie bei 200 Systemtypelementen.

Die Laufzeitmessungen haben gezeigt, dass die Laufzeit mit einem zunehmenden Umfang der SoS-Zielspezifikation ansteigt. Aber auch bei relativ großen Spezifikationen mit über 3000 Modellelementen können innerhalb kürzester Zeit Sichten generiert werden. Dabei scheint die Laufzeit sowohl von der Anzahl der Elemente im Zielmodell als auch von der Anzahl der Elemente im SoS-Modell und der Anzahl der X-Links abzuhängen. Demzufolge

stellt der Zeitaufwand für die Generierung vieler Sichten kein Hemmnis bei der Nutzung des vorgestellten Ansatzes dar.

9.3 Diskussion

Der vorgestellte Ansatz wurde an einem Fallbeispiel aus der Robotik-Industrie angewendet. Außerdem wurde der Ansatz prototypisch implementiert. Die Anwendung am Fallbeispiel und die prototypische Implementierung zeigen, dass der vorgestellte Ansatz grundsätzlich anwendbar ist. Für das Fallbeispiel konnte eine SoS-Zielspezifikation bestehend aus einem Zielmodell, einem SoS-Modell und X-Links erstellt werden. Außerdem konnte nachgewiesen werden, dass sich aus dieser SoS-Zielspezifikation sowohl Zielsichten als auch SoS-Konfigurationssichten ableiten lassen. Anhand der prototypischen Implementierung wurde deutlich, dass die Generierung von Sichten automatisierbar ist. Laufzeitmessungen ergaben, dass die automatisierte Sichtenerstellung in akzeptabler Zeit erfolgt, sodass dies kein Hindernis für die Nutzung des Ansatzes darstellt.

Grundsätzlich treten bei Fallbeispielen Validitätsgefährdungen im Bereich der externen Validität auf. Insbesondere stellt sich die Frage, ob die Erkenntnisse generalisierbar sind. D. h., sind die Ergebnisse auf andere Fallbeispiele sowie auf einen industriellen Kontext übertragbar? Um ein Mindestmaß an Industrierelevanz sicherzustellen, wurde das Fallbeispiel in Zusammenarbeit mit Experten aus der Robotik-Industrie entwickelt. Dennoch bestehen weiterhin Gefährdungen bzgl. der Übertragbarkeit, da Untersuchungen am Fallbeispiel immer nur auf dieses eine Fallbeispiel abzielen. Möglicherweise existieren demnach auch andere Systems-of-Systems, für die der Ansatz in seiner jetzigen Form nicht geeignet ist. Die prototypische Implementierung hat gezeigt, dass Sichten in akzeptabler Zeit generierbar sind. Bei der Ermittlung der Laufzeit für verschieden große SoS-Zielspezifikationen kann es jedoch zu Messfehlern (bspw. durch parallele Prozesse) gekommen sein. Jedoch ist durch die prototypische Implementierung deutlich geworden, dass der vorgestellte Ansatz realisierbar ist.

Insgesamt erlauben die Ergebnisse die Schlussfolgerung, dass der vorgeschlagene Ansatz prinzipiell anwendbar ist. Somit kann die Evaluationsfragestellung E1 als erfüllt angesehen werden.

10 Kapitel
Evaluation der Vorteilhaftigkeit

Zur Bewertung der Vorteilhaftigkeit wurde ein kontrolliertes Experiment durchgeführt. Das Experiment zeigt, dass die Nutzung der Sichten beim Review im Vergleich zur Nutzung der SoS-Zielspezifikation Vorteile bietet.

Der Aufbau dieses Kapitels orientiert sich an den Empfehlungen von Wohlin et al. (2012) für Berichte von kontrollierten Experimenten. In Abschnitt 10.1 werden die Entscheidungen erläutert, die im Rahmen der Experimentplanung getroffen wurden. Abschnitt 10.2 berichtet über die Durchführung des Experiments. In Abschnitt 10.3 werden die Ergebnisse des Experiments vorgestellt. Abschnitt 10.4 bietet eine Ergänzung, die verschiedene Faktoren untersucht, um einen genaueren Einblick in die Ergebnisse zu erhalten. Abschnitt 10.5 diskutiert die Bedeutung der Ergebnisse für die Evaluation des Ansatzes.

10.1 Experimentplanung

Dieser Abschnitt bietet einen Einblick in den Aufbau des Experiments. Vorgestellt werden das Ziel, die Teilnehmer, das Experimentmaterial, die Aufgabe, die Variablen und Hypothesen, der Experimentaufbau, der Ablauf sowie die Auswertung.

10.1.1 Ziel

Das Ziel des Experiments ist es zu prüfen, ob die generierten Sichten den Reviewer bei der Validierung von SoS-Zielspezifikationen unterstützen. Dazu werden die Effektivität, die Effizienz, die Selbstsicherheit der Experimentteilnehmer und die subjektive Unterstützungsfähigkeit der Sicht mit der SoS-Zielspezifikation verglichen.

10.1.2 Teilnehmer

Als Experimentteilnehmer wurden Studenten ausgewählt, da neuere Forschungsergebnisse gezeigt haben, dass der Einsatz von studentischen Teilnehmern signifikantere Schlussfolgerungen unterstützt und als faireres Experiment zwischen dem zu evaluierenden Ansatz und dem Vergleichsansatz anzusehen ist als der Einsatz von Fachleuten aus der Industrie (Salman

et al. 2015; Siegmund et al. 2015). Der Experimentaufbau und die Experimentergebnisse wurden jedoch mit Industriepartnern besprochen, um die Verallgemeinerbarkeit zu gewährleisten. Das Experiment wurde mit Studenten im Rahmen zweier Universitätskurse zum Thema Requirements Engineering durchgeführt – und zwar in Lerneinheiten zur konzeptionellen Modellierung von Anforderungen. Die Teilnehmer studieren hauptsächlich 'Angewandte Informatik' (mit Schwerpunkt Software Engineering) oder 'Wirtschaftsinformatik'. 142 Bachelorstudenten und 39 Masterstudenten haben an dem Experiment teilgenommen. Die Studenten wurden innerhalb der Kurse rekrutiert. Es wurden keine Boni in Bezug auf die Abschlussprüfung des Kurses vergeben, um soziale Risiken für die Validität zu vermeiden. Insofern ist die Rekrutierungsstrategie als willkürliche Stichprobe zu betrachten.

10.1.3 Experimentmaterial

Als Quelle für das Versuchsmaterial diente die in Abschnitt 9.1 vorgestellte Beispielspezifikation aus dem Bereich Robotik. Die in Abbildung 9.3 dargestellte Spezifikation besteht aus einem Zielmodell, einem SoS-Modell und X-Links. Zwei Sichten auf das Zielmodell und zwei Sichten auf das SoS-Modell wurden aus dieser SoS-Zielspezifikation abgeleitet. Jeweils eine dieser Sichten wird in den Abbildungen 9.4 und 9.5 dargestellt. Um die Verallgemeinerbarkeit auf einen realen industriellen Entwicklungsprozess zu gewährleisten, wurde das Experimentmaterial in enger Zusammenarbeit mit Industrieexperten eines führenden europäischen Unternehmens im Bereich Robotik vorbereitet.

10.1.4 Aufgabe

Bei dem Experiment werden den Teilnehmern Modelle vorgelegt, bei denen es sich entweder um eine SoS-Zielspezifikation bestehend aus einem Zielmodell, SoS-Modell und X-Links handelt oder um eine Sicht (Zielsicht oder SoS-Konfigurationssicht). Zu jedem Modell werden natürlichsprachliche Aussagen bezogen auf die Erfüllbarkeit von Zielen bei bestimmten SoS-Konfigurationen angezeigt. Die Teilnehmer sind aufgefordert, für jede der Aussagen anzugeben, ob sie in dem jeweils angezeigten Modell dargestellt sind oder nicht. Außerdem geben die Teilnehmer bei jeder Aussage an, wie sicher sie sich bzgl. ihrer Antwort sind.

10.1.5 Variablen und Hypothesen

Als unabhängige Variable wird das Darstellungsformat des Review-Artefakts mit den folgenden Ausprägungen untersucht:

- die ***Sichten*** (Zielsicht und SoS-Konfigurationssicht)
- die ***SoS-Zielspezifikation*** (Zielmodell, SoS-Modell, X-Links)

Abhängige Variablen sind:

- ***Effektivität***: der Anteil der richtigen Antworten.
- ***Effizienz***: der durchschnittliche Zeitaufwand pro richtiger Antwort.
- ***Selbstgewissheit***: die durchschnittliche Bewertung auf einer fünfstufigen semantischen Differential-Skala, die die Selbstgewissheit der Teilnehmer widerspiegelt, ob die gegebene Antwort richtig ist.
- ***Subjektive Unterstützungsfähigkeit***: die durchschnittliche Bewertung auf einer fünfstufigen semantischen Differential-Skala, die die Unterstützungfähigkeit des Darstellungsformats widerspiegelt.

Ausgehend von dem Ziel des Experiments, den unabhängigen und den abhängigen Variablen können die folgenden Null- und Alternativhypothesen des Experiments formuliert werden.

Hypothesen zur ***Effektivität***:

- $H_1 0$ Das Darstellungsformat hat keinen signifikanten Einfluss auf die Effektivität des Reviews.
- $H_1 a$ Die Sichten unterstützen einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
- $H_1 b$ Die SoS-Zielspezifikation unterstützt einen signifikant effektiveren Review im Vergleich zu den Sichten.

Hypothesen zur ***Effizienz***:

- $H_2 0$ Das Darstellungsformat hat keinen signifikanten Einfluss auf die Effizienz des Reviews.
- $H_2 a$ Die Sichten unterstützen einen signifikant effizienteren Review im Vergleich zur SoS-Zielspezifikation.
- $H_2 b$ Die SoS-Zielspezifikation unterstützt einen signifikant effizienteren Review im Vergleich zu den Sichten.

Hypothesen zur **Selbstgewissheit**:

- $H_3 0$ Das Darstellungsformat hat keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.
- $H_3 a$ Die Sichten unterstützen eine signifikant höhere Selbstgewissheit des Reviewers im Vergleich zur SoS-Zielspezifikation.
- $H_3 b$ Die SoS-Zielspezifikation unterstützt eine höhere Selbstgewissheit des Reviewers im Vergleich zu den Sichten.

- $H_3^B 0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.

Hypothesen zur ***subjektiven Unterstützungsfähigkeit***:

- $H_4 0$ Das Darstellungsformat hat keinen signifikanten Einfluss auf die subjektive Unterstützungsfähigkeit.
- $H_4 a$ Die Sichten führen zu einer signifikant höheren subjektiven Unterstützungsfähigkeit im Vergleich zur SoS-Zielspezifikation.
- $H_4 b$ Die SoS-Zielspezifikation führt zu einer signifikant höheren subjektiven Unterstützungsfähigkeit im Vergleich zu den Sichten.

10.1.6 Experimentaufbau

Das Experiment wurde mithilfe eines Online-Fragebogens[8] über https://www.soscisurvey.de durchgeführt. Das Experiment war auf eine Dauer von etwa 30 Minuten ausgelegt, um die Abbruchwahrscheinlichkeit aufgrund von Interessenverlust zu minimieren. Für das Experiment wurde ein Within-Subject-Design (Wohlin 2012) verwendet. Dieser Aufbau wurde auf der Grundlage bewährter Praktiken gewählt (Daun et al. 2021).

10.1.7 Ablauf

Jeder Teilnehmer führte einen Review der SoS-Zielspezifikation und der beiden Sichten durch. Die Reihenfolge der Review-Artefakte mit den verschiedenen Darstellungsformen wurde für jeden Teilnehmer randomisiert. Bei jedem Review-Artefakt wurden den Teilnehmern Aussagen vorgelegt, die sich

a) auf erfüllbare Ziele von ausgewählten SoS-Konfigurationen oder

b) auf SoS-Konfigurationen, die ausgewählte Ziele erfüllen können,

bezogen. Nachfolgend wird jeweils ein Beispiel aus diesen Aussagen gezeigt.

[8] Ein Ausdruck des Online-Fragebogens ist im Anhang zu finden.

Beispiel für Aussagen bzgl. erfüllbarer Ziele:

A SoS-configuration consisting of:

- one charging station type I
- one charging station type III
- three robots type II
- three robots type V
- one robot type VIII
- two pick-up stations type II
- two drop-off stations of type II

can move a heavy load.

Beispiel für Aussagen bzgl. SoS-Konfigurationen:

The goals/task pick-ups/drop-offs at different heights and move heavy load can be fulfilled by a SoS-configuration consisting of:

- four charging stations type I
- three robots type III
- one robot type VI
- three robots type VIII
- two pick-up stations type III
- one drop-off station type III

Die Teilnehmer mussten entscheiden, ob die Aussage gemäß dem gezeigten Modell wahr ist oder nicht. Insgesamt wurden den Teilnehmern vier Sichten (zwei Zielsichten und zwei SoS-Konfigurationssichten) und vier Mal die SoS-Zielspezifikation angezeigt. Nach dem Review wurde jeder Teilnehmer zur subjektiven Unterstützungsfähigkeit der Modelle befragt. Danach wurden einige persönlichen Daten erhoben.

10.1.8 Auswertung

Zur Prüfung der Hypothesen zu Effektivität, Effizienz und Selbstgewissheit werden t-Tests für abhängige Stichproben durchgeführt. Die Analyse der Daten für subjektive Unterstützungsfähigkeit erfolgt anhand eines Einstichproben-t-Tests. Aufgrund des großen Umfangs der Stichproben kommt der zentrale Grenzwertsatz zum Tragen, sodass auch bei Abweichungen von der Normalverteilung parametrische Tests anwendbar sind (Field 2013).

10.2 Experimentdurchführung

Die folgenden Abschnitte beschreiben die Experimentdurchführung.

10.2.1 Vorbereitung

Um die Qualität des Experiments sicherzustellen, wurde ein Vortest (Wohlin 2012) mit acht Teilnehmern durchgeführt. Dadurch wurde gewährleistet, dass die automatisierte Randomisierung funktioniert und nicht allen Teilnehmern die Modelle in der gleichen Reihenfolge angezeigt werden. Die Ergebnisse des Vortests sind nicht in die Auswertung des Experiments eingeflossen. Die Teilnehmer des Experiments wurden nicht detailliert in das Experiment eingewiesen, um die Ergebnisse nicht zu beeinflussen.

10.2.2 Abweichungen

Im Rahmen der Experimentdurchführung sind keine Abweichungen aufgetreten. Die Teilnehmer hatten für die Mitwirkung an dem Experiment jeweils fünf Tage Zeit. Die meisten Teilnehmer haben das Experiment innerhalb von 30 Minuten durchgeführt. Es erfolgte keine Datenbereinigung. Alle erhobenen Datensätze wurden ausgewertet.

10.3 Experimentergebnisse

Die folgenden Abschnitte stellen die Ergebnisse der Experimentauswertung vor. Dies schließt sowohl die deskriptiven Statistiken als auch die Ergebnisse der Hypothesentests ein.

10.3.1 Deskriptive Statistiken

Dieser Abschnitt befasst sich mit den deskriptiven Statistiken für die abhängigen Variablen Effektivität, Effizienz, Selbstgewissheit und subjektive Unterstützungsfähigkeit. Tabelle 10.1 zeigt die deskriptiven Statistiken für die Effektivität, Effizienz, Selbstgewissheit und subjektive Unterstützungsfähigkeit von Sichten und SoS-Zielspezifikationen.

Effektivität

Die Effektivität des Reviews ist bei der Nutzung der Sichten ($M = 67,19\%, SD = 19.31\%$) im Durchschnitt höher als bei der Nutzung der SoS-Zielspezifikation ($M = 61,12\%, SD = 16.21\%$). Die Verteilung wird in Abbildung 10.1 durch Boxplots illustriert. Die Boxplots zeigen, dass die Effektivität der meisten Teilnehmer bei der SoS-Zielspezifikation unter 70% und bei den Sichten unter 90% liegt. In Abbildung 10.2 illustriert ein Streudiagramm die Effektivität der Teilnehmer beim Review von Sichten und SoS-Zielspezifikationen. Die Punkte symbolisieren die Teilnehmer. Punkte oberhalb der Diagonalen repräsentieren Teilnehmer, die bei der Nutzung der Sichten eine höhere Effektivität aufwiesen als bei der Nutzung der SoS-Zielspezifikation.

Tabelle 10.1: Deskriptive Statistiken

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
Effektivität	Sichten	181	67,19%	64,36%	70,03%	62,07%	19,31%	24,14%	100,00%	34,48%
	SoS-Zielspezifikation	181	61,12%	58,74%	63,49%	58,62%	16,21%	24,14%	96,55%	22,41%
Effizienz	Sichten	181	29,59	25,21	33,98	23,43	29,88	2,47	207,06	30,23
	SoS-Zielspezifikation	181	35,13	26,80	43,47	17,53	56,83	2,32	525,44	35,78
Selbstgewissheit	Sichten	181	3,49	3,35	3,63	3,43	0,97	1,00	5,00	1,37
	SoS-Zielspezifikation	181	3,38	3,23	3,53	3,31	1,03	1,00	5,00	1,37
Subjektive Unterstützungsfähigkeit		181	3,18	3,05	3,31	3,00	0,88	1,00	5,00	1,10

Wie zu erkennen ist, waren die meisten Teilnehmer bei der Nutzung der Sichten effektiver als bei der Nutzung der SoS-Zielspezifikation.

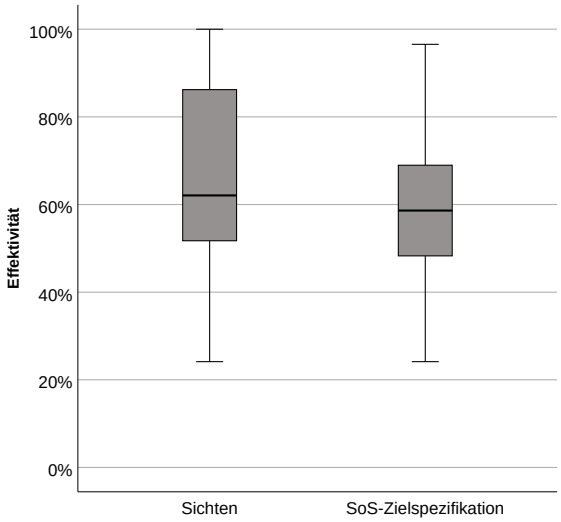


Abbildung 10.1: Effektivität – Boxplots

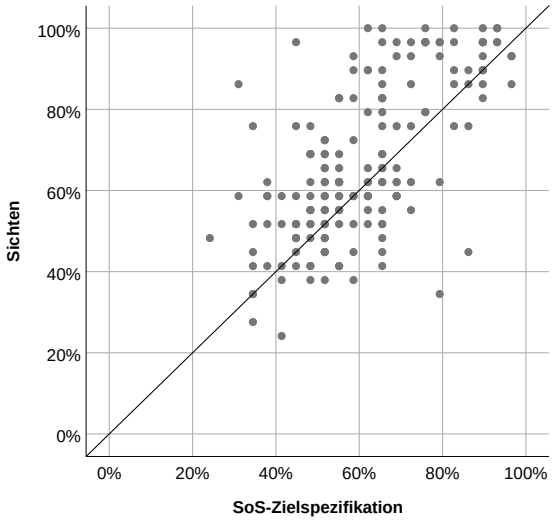


Abbildung 10.2: Effektivität – Streudiagramm

Effizienz

Die Effizienz des Reviews ist im Durchschnitt bei der Nutzung der Sichten ($M = 29,21, SD = 29,88$) geringer als bei der Nutzung der SoS-Zielspezifikation ($M = 35,13, SD = 56,83$). Effizienz beschreibt die Dauer in Sekunden, die die Teilnehmer durchschnittlich für jede richtige Antwort benötigt haben. Die Verteilung ist in Abbildung 10.3 durch Boxplots illustriert. Die Boxplots zeigen, dass es mehrere Ausreißer gibt, die durchschnittlich sehr lange für eine richtige

Antwort gebraucht haben. In Abbildung 10.4 illustriert ein Streudiagramm die Effizienz der Teilnehmer beim Review von Sichten und SoS-Zielspezifikationen. Die Punkte symbolisieren die Teilnehmer. Punkte oberhalb der Diagonalen repräsentieren Teilnehmer, die bei der Nutzung der Sichten eine höhere Effizienz aufwiesen als bei der Nutzung der SoS-Zielspezifikation. Wie zu erkennen ist, erzielten die meisten Teilnehmer eine ähnliche Effizienz bei der Nutzung der Sichten wie bei der Nutzung der SoS-Zielspezifikation. Jedoch war die Effizienz einiger Teilnehmer beim Review der SoS-Zielspezifikation deutlich niedriger als beim Review der Sichten.

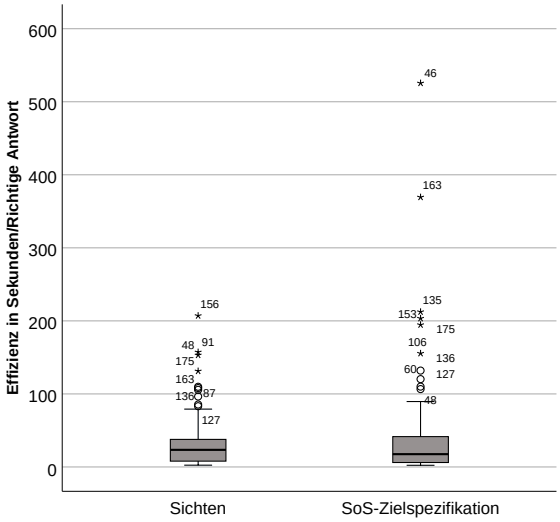


Abbildung 10.3: Effizienz – Boxplots

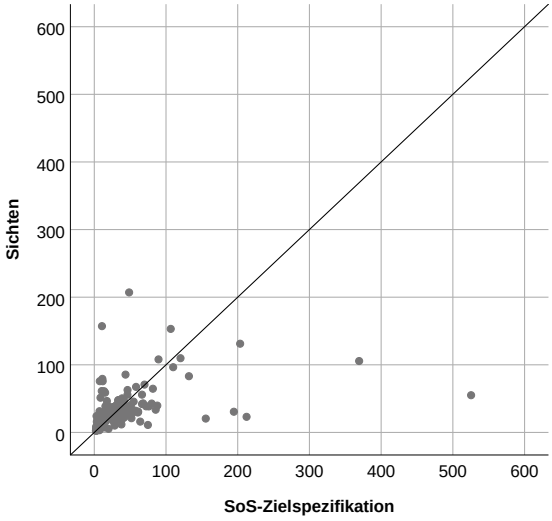


Abbildung 10.4: Effizienz – Streudiagramm

Selbstgewissheit

Die Selbstgewissheit wurde auf einer fünfstufigen semantischen Differential-Skala gemessen, wobei eins die geringste und fünf die höchste Selbstgewissheit ausdrückt. Die Selbstgewissheit des Reviewers ist im Durchschnitt bei der Nutzung der Sichten ($M = 3,49, SD = 0,97$) höher als bei der Nutzung der SoS-Zielspezifikation ($M = 3,38, SD = 1,03$). Die Verteilung ist in Abbildung 10.5 durch Boxplots illustriert. Die Boxplots zeigen, dass die durchschnittliche Selbstgewissheit sowohl bei der Nutzung der Sichten als auch bei Nutzung der SoS-Zielspezifikation über dem Erwartungswert von drei liegt. In Abbildung 10.6 illustriert ein Streudiagramm die Selbstgewissheit der Teilnehmer beim Review von Sichten und SoS-Zielspezifikationen. Die Punkte symbolisieren die Teilnehmer. Punkte oberhalb der Diagonalen repräsentieren Teilnehmer, die bei der Nutzung der Sichten eine höhere Selbstgewissheit aufwiesen als bei der Nutzung der SoS-Zielspezifikation. Wie zu erkennen ist, lassen die meisten Teilnehmer eine ähnliche Selbstgewissheit bei der Nutzung der Sichten erkennen wie bei der Nutzung der SoS-Zielspezifikation.

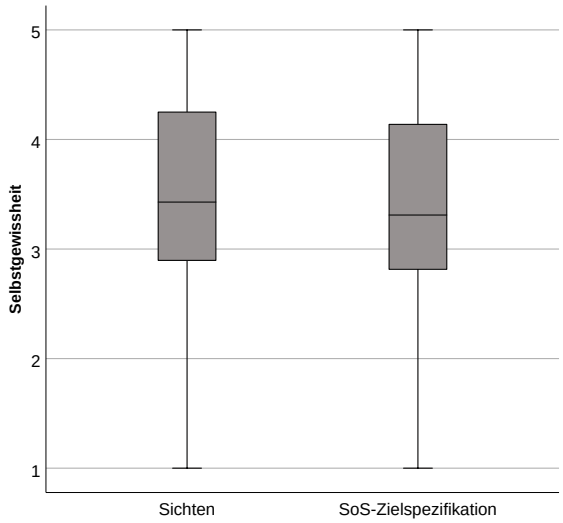


Abbildung 10.5: Selbstgewissheit – Boxplots

Subjektive Unterstützungsfähigkeit

Die subjektive Unterstützungsfähigkeit wurde ebenfalls auf einer fünfstufigen semantischen Differential-Skala gemessen. Die Teilnehmer bewerten hier die Nutzung der Sichten gegen die Nutzung der SoS-Zielspezifikation, wobei ein niedriger Wert eine Bevorzugung der SoS-Zielspezifikation und ein hoher Wert eine Bevorzugung der Sichten bedeutet. Die Teilnehmer haben durchschnittlich die Sicht als subjektiv unterstützender bewertet ($M = 3,18, SD = 0,88$).

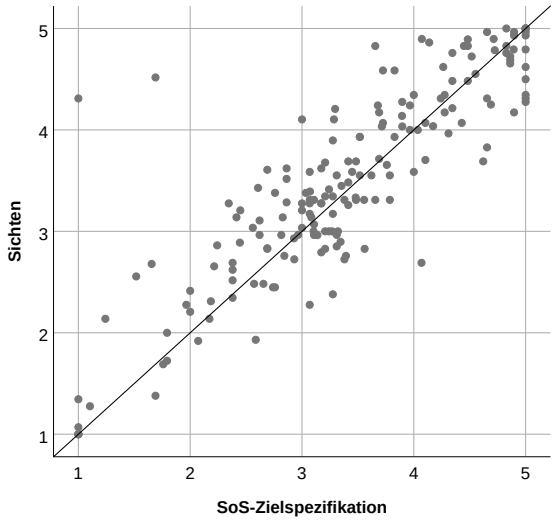


Abbildung 10.6: Selbstgewissheit – Streudiagramm

10.3.2 Hypothesentests

Die Ergebnisse der Hypothesentests für Effektivität, Effizienz und Selbstgewissheit werden in Tabelle 10.2 gezeigt.

Effektivität

Der t-Test verdeutlicht, dass die Effektivität des Reviews bei der Nutzung der Sichten signifikant höher ist als bei der Nutzung der SoS-Zielspezifikation $t(180) = 5,55, p < 0,001$. Die Alternativhypothese H_1a wird angenommen.

Effizienz

Des Weiteren zeigt der t-Test, dass es bezüglich der Effizienz des Reviews keinen signifikanten Unterschied zwischen der Nutzung der Sichten und der SoS-Zielspezifikation gibt $t(180) = -1,44, p = 0,15$. Die Nullhypothese H_20 kann nicht verworfen werden.

Selbstgewissheit

Außerdem legt der t-Test nahe, dass die Selbstgewissheit des Reviewers bei Nutzung der Sichten signifikant höher ist als bei der Nutzung der SoS-Zielspezifikation $t(180) = 2,76, p = 0,01$. Die Alternativhypothese H_3a wird angenommen.

Subjektive Unterstützungsfähigkeit

Die subjektive Unterstützungsfähigkeit wurde auf einer fünfstufigen semantischen Differential-Skala gemessen. Die Teilnehmer bewerten hier die Nutzung der Sichten gegen die Nutzung der

Tabelle 10.2: Ergebnisse der t-Tests für abhängige Stichproben

	Gepaarte Differenzen							
				95% Konfidenzintervall der Differenz				
	Mittelwert	Std.-Abweichung	Standardfehler des Mittelwertes	Untere	Obere	T	df	Sig. (2-seitig)
Effektivität	6%	15%	1%	4%	8%	5,55	180,0	<0,001
Effizienz	-5,54	51,84	3,85	-13,14	2,07	-1,44	180,0	0,15
Selbstgewissheit	0,11	0,53	0,04	0,03	0,19	2,76	180,0	0,01

SoS-Zielspezifikation, wobei ein niedriger Wert eine Bevorzugung der SoS-Zielspezifikation und ein hoher Wert eine Bevorzugung der Sichten bedeutet. Auf einer Skala von eins bis fünf stellt der Wert drei den Erwartungswert für Gleichwertigkeit dar. Das Ergebnis des t-Tests gegen den Erwartungswert drei wird in Tabelle 10.3 gezeigt. Die Sichten wurden als signifikant subjektiv unterstützender bewertet als die SoS-Zielspezifikation $t(180) = 2,69, p = 0,01$. Die Alternativhypothese H_4a wird angenommen.

Tabelle 10.3: Ergebnis des t-Tests gegen Erwartungswert

					95% Konfidenzintervall der Differenz	
	T	df	Sig. (2-seitig)	Mittlere Differenz	Untere	Obere
Subjektive Unterstützungsfähigkeit	2,69	180	0,008	0,176	0,05	0,31

Tabelle 10.4 gibt eine Übersicht über die angenommenen Alternativ- und nicht verworfenen Nullhypothesen.

Tabelle 10.4: Übersicht über die Ergebnisse der Hypothesentests

Hypothese	p	Bedeutung
H_1a	<0,001	Die Sichten unterstützen einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
H_20	0,15	Das Darstellungsformat hat keinen signifikanten Einfluss auf die Effizienz des Reviews.
H_3a	0,01	Die Sichten unterstützen eine signifikant höhere Selbstgewissheit des Reviewers im Vergleich zur SoS-Zielspezifikation.
H_4a	0,008	Die Sichten führen zu einer signifikant höheren subjektiven Unterstützungsfähigkeit im Vergleich zur SoS-Zielspezifikation.

10.4 Erweiterte Experimentauswertung

Die Auswertung des Experiments hat gezeigt, dass die Nutzung von Sichten effektiver ist, die Selbstgewissheit erhöht und als unterstützender angesehen wird. Um dies zu bestätigen, wurde die Nutzung der Sichten mit der Nutzung der SoS-Zielspezifikation verglichen. Unklar ist jedoch bislang, ob dies allgemein gilt oder ob es Faktoren gibt, die dazu führen, dass die Nutzung der SoS-Zielspezifikation zu bevorzugen ist. Im Rahmen einer genaueren Untersuchung werden in diesem Abschnitt zwei Faktoren berücksichtigt, die möglicherweise Einfluss nehmen:

- Art der Sicht (Zielsicht oder SoS-Konfigurationssicht)
- Erfahrungsniveau (Bachelorstudent oder Masterstudent)

Zur Feststellung, ob sowohl die Nutzung der Zielsicht als auch die Nutzung der SoS-Konfigurationssicht vorteilhaft sind, wird das Experiment in zwei Teile differenziert. Der erste Teil betrachtet die Fragen zur Zielsicht im Vergleich zu dem diesbezüglichen Teil für die SoS-Zielspezifikation. Der zweite Teil betrachtet die Fragen zur SoS-Konfigurationssicht im Vergleich zu dem jeweiligen Teil für die SoS-Zielspezifikation.

Zum Vergleich des Erfahrungsniveaus werden die Ergebnisse der Bachelorstudenten mit denjenigen der Masterstudenten verglichen. Dadurch zeigt sich nicht nur, welche Gruppe bessere Reviews durchführt, sondern auch, ob es Interaktionseffekte[9] gibt, die dazu führen, dass für die unterschiedlichen Gruppen unterschiedliche Darstellungsformen besser geeignet sind.

10.4.1 Zielsicht

Im folgenden Abschnitt werden die Hypothesen, das Verfahren zur Auswertung sowie die Ergebnisse für den Vergleich der Zielsicht mit der SoS-Zielspezifikation eingehend betrachtet.

10.4.1.1 Hypothesen für die Zielsicht

Für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit wird jeweils eine Nullhypothese für die beiden Haupteffekte (Darstellungsformat und Erfahrungsniveau) und die beiden gerichteten Alternativhypothesen definiert. Außerdem wird für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit eine Nullhypothese für den Interaktionseffekt zwischen Darstellungsformat und Erfahrungsniveau sowie eine ungerichtete Alternativhypothese für das Vorhandensein einer Interaktion definiert. Ergänzend ergeben sich die folgenden Hypothesen:

[9] Interaktionseffekte beschreiben Effekte, die auf Wechselbeziehungen zwischen unabhängigen Variablen zurückzuführen sind.

Hypothesen zur ***Effektivität***:

- $H_{1.1}0$ Das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen [10] keinen signifikanten Einfluss auf die Effektivität des Reviews.
- $H_{1.1}a$ Die Zielsicht unterstützt einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
- $H_{1.1}b$ Die SoS-Zielspezifikation unterstützt einen signifikant effektiveren Review bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen im Vergleich zur Zielsicht.
- $H^B_{1.1}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Effektivität des Reviews.
- $H^B_{1.1}a$ Bachelorstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Masterstudenten.
- $H^B_{1.1}b$ Masterstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Bachelorstudenten.
- $H^I_{1.1}0$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.
- $H^I_{1.1}a$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.

Hypothesen zur ***Effizienz***:

- $H_{1.2}0$ Das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Effizienz des Reviews.
- $H_{1.2}a$ Die Zielsicht unterstützt einen signifikant effizienteren Review im Vergleich zur SoS-Zielspezifikation.

[10] Die SoS-Zielspezifikation selbst war bei dem Vergleich mit der Zielsicht und der SoS-Konfigurationssicht identisch. Allerdings unterschieden sich die natürlichsprachlichen Aussagen. Für den Vergleich mit der Zielsicht wurden Aussagen vorgegeben, die sich anhand der Zielsicht beantworten lassen, d. h. Aussagen bzgl. der Zielerfüllung von ausgewählten SoS-Konfigurationen. Beim Vergleich mit der SoS-Konfigurationssicht wurden Aussagen vorgegeben, die sich anhand der SoS-Konfigurationssicht beantworten lassen, d. h. Aussagen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können.

- $H_{1.2}b$ Die SoS-Zielspezifikation unterstützt einen signifikant effizienteren Review bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen im Vergleich zur Zielsicht.
- $H_{1.2}^{B}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Effizienz des Reviews.
- $H_{1.2}^{B}a$ Bachelorstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Masterstudenten.
- $H_{1.2}^{B}b$ Masterstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effizientere Reviews durch als Bachelorstudenten.
- $H_{1.2}^{I}0$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.
- $H_{1.2}^{I}a$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.

Hypothesen zur **Selbstgewissheit**:

- $H_{1.3}0$ Das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.
- $H_{1.3}a$ Die Zielsicht unterstützt eine signifikant höhere Selbstgewissheit des Reviewers im Vergleich zur SoS-Zielspezifikation.
- $H_{1.3}b$ Die SoS-Zielspezifikation unterstützt eine höhere Selbstgewissheit des Reviewers bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen im Vergleich zur Zielsicht.
- $H_{1.3}^{B}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.
- $H_{1.3}^{B}a$ Bachelorstudenten haben bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen eine höhere Selbstgewissheit als Masterstudenten.
- $H_{1.3}^{B}b$ Masterstudenten haben bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen eine höhere Selbstgewissheit als Bachelorstudenten.

- $H^I_{1.3}0$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.
- $H^I_{1.3}a$ Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.

10.4.1.2 Auswertung

Zur Prüfung der Hypothesen zu Effektivität, Effizienz und Selbstgewissheit wird eine gemischte Varianzanalyse (mixed ANOVA) durchgeführt. Auf diese Weise wird sowohl der Within-Subject-Faktor Darstellungsformat als auch der Between-Subject-Faktor Erfahrungsniveau untersucht.

10.4.1.3 Deskriptive Statistiken

In diesem Abschnitt wird die deskriptive Statistik für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit für die Zielsicht präsentiert. Die Zielsicht zeigt die für ausgewählte SoS-Konfigurationen als erfüllbar spezifizierten intentionalen Elemente eines Zielmodells an. Um den Nutzen der Zielsicht zu untersuchen, wurde dieser mit der Nutzung der SoS-Zielspezifikation verglichen.

Effektivität

Tabelle 10.5 zeigt die deskriptive Statistik für die Effektivität der Zielsicht. Aus den Daten kann abgeleitet werden, dass bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen die Nutzung der Zielsicht $(M = 69,76\%, SD = 22,27\%)$ eine höhere Effektivität aufweist als die Nutzung der SoS-Zielspezifikation $(M = 62,84\%, SD = 18,00\%)$. Außerdem zeigt sich, dass Masterstudenten sowohl bei der Nutzung der Zielsicht als auch bei der Nutzung der SoS-Zielspezifikation effektiver sind als Bachelorstudenten bei der Nutzung des jeweiligen Darstellungsformats. Abbildung 10.7 vergleicht die Verteilung für die Effektivität von Zielsicht und SoS-Zielspezifikation bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen. Es ist erkennbar, dass bei Masterstudenten der Unterschied in der Effektivität zwischen der Zielsicht und der SoS-Zielspezifikation stärker ausgeprägt ist als bei Bachelorstudenten. Das bedeutet, dass die Nutzung von Zielsichten effektiver ist als die Nutzung der SoS-Zielspezifikation und dass der Unterschied für erfahrenere Reviewer größer ist als für weniger erfahrene.

Effizienz

Tabelle 10.6 gibt die deskriptive Statistik für die Effizienz der Zielsicht wieder. Die Daten zeigen, dass die Nutzung der Zielsicht $(M = 22,07, SD = 25,62)$ im arithmetischen Mittel

Tabelle 10.5: Deskriptive Statistik - Effektivität der Zielsicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
Zielsicht	Alle Teilnehmer	181	69,76%	66,49%	73,02%	63,16%	22,27%	26,32%	100,00%	42,11%
	Bachelorstudenten	142	64,79%	61,32%	68,26%	57,89%	20,93%	26,32%	100,00%	32,89%
	Masterstudenten	39	87,85%	82,29%	93,42%	100,00%	17,17%	47,37%	100,00%	26,32%
SoS-Ziel-	Alle Teilnehmer	181	62,84%	60,20%	65,48%	63,16%	18,00%	15,79%	100,00%	26,32%
spezifikation	Bachelorstudenten	142	60,38%	57,44%	63,31%	57,89%	17,68%	15,79%	100,00%	21,05%
	Masterstudenten	39	71,79%	66,48%	77,11%	73,68%	16,40%	36,84%	94,74%	21,05%

Tabelle 10.6: Deskritpive Statistik - Effizienz der Zielsicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
Zielsicht	Alle Teilnehmer	181	22,07	18,32	25,83	16,69	25,62	1,92	180,80	18,69
	Bachelorstudenten	142	19,57	15,86	23,28	12,88	22,34	1,92	150,05	18,13
	Masterstudenten	39	31,19	20,21	42,17	21,26	33,87	2,08	180,80	14,12
SoS-Ziel-	Alle Teilnehmer	181	24,45	19,42	29,48	13,73	34,32	1,92	350,44	29,20
spezifikation	Bachelorstudenten	142	22,69	16,52	28,86	8,72	37,19	2,33	350,44	22,83
	Masterstudenten	39	30,87	24,43	37,30	31,19	19,85	1,92	92,47	20,65

Tabelle 10.7: Deskriptive Statistik - Selbstgewissheit bei der Zielsicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
Zielsicht	Alle Teilnehmer	181	3,51	3,36	3,66	3,53	1,04	1,00	5,00	1,58
	Bachelorstudenten	142	3,37	3,19	3,54	3,32	1,04	1,00	5,00	1,37
	Masterstudenten	39	4,04	3,77	4,31	4,32	0,84	2,21	5,00	1,16
SoS-Ziel-	Alle Teilnehmer	181	3,43	3,27	3,59	3,42	1,09	1,00	5,00	1,51
spezifikation	Bachelorstudenten	142	3,27	3,09	3,45	3,21	1,06	1,00	5,00	1,32
	Masterstudenten	39	4,01	3,69	4,34	4,32	1,01	1,53	5,00	1,47

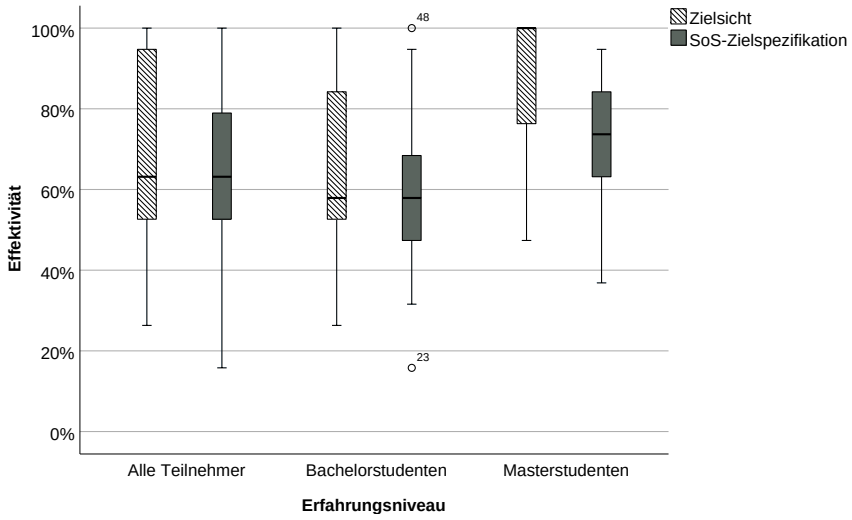


Abbildung 10.7: Effektivität der Zielsicht

effizienter ist als die Nutzung der SoS-Zielspezifikation ($M = 24,45, SD = 34,32$) bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen. Bei Betrachtung des Medians zeichnet sich jedoch ein gegenteiliges Bild ab. Auffallend ist hier vor allem die höhere Effizienz der Bachelorstudenten ($Mdn = 12,88$ für die Zielsicht und $Mdn = 8,72$ für die SoS-Zielspezifikation) verglichen mit den Masterstudenten ($Mdn = 21,26$ für die Zielsicht und $Mdn = 31,19$ für die SoS-Zielspezifikation). Abbildung 10.8 illustriert die Verteilung für die abhängige Variable Effizienz von Zielsicht und SoS-Zielspezifikation bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen. Es ist erkennbar, dass bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen die Effizienz sowohl bei der Nutzung der Zielsicht als auch bei der Nutzung der SoS-Zielspezifikation einer großen Streuung unterliegt. Insbesondere gibt es viele Ausreißer mit geringer Effizienz (d. h. Teilnehmer, die eine hohe Anzahl an Sekunden pro richtiger Antwort gebraucht haben).

Selbstgewissheit

Tabelle 10.7 kann die deskriptive Statistik für die Selbstgewissheit bei der Nutzung der Zielsicht entnommen werden. Die Daten zeigen, dass bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen die durchschnittliche Selbstgewissheit bei der Nutzung der Zielsicht ($M = 3,51, SD = 1,04$) höher ist als bei der Nutzung der SoS-Zielspezifikation ($M = 3,43, SD = 1,09$). Allerdings scheint dieser Unterschied klein zu sein. Abbildung 10.9 illustriert die Verteilung für die abhängige Variable Selbstgewissheit bei der Nutzung der Zielsicht und der SoS-Zielspezifikation bei Fragestellungen bzgl. der Zielerfüllung ausgewählter

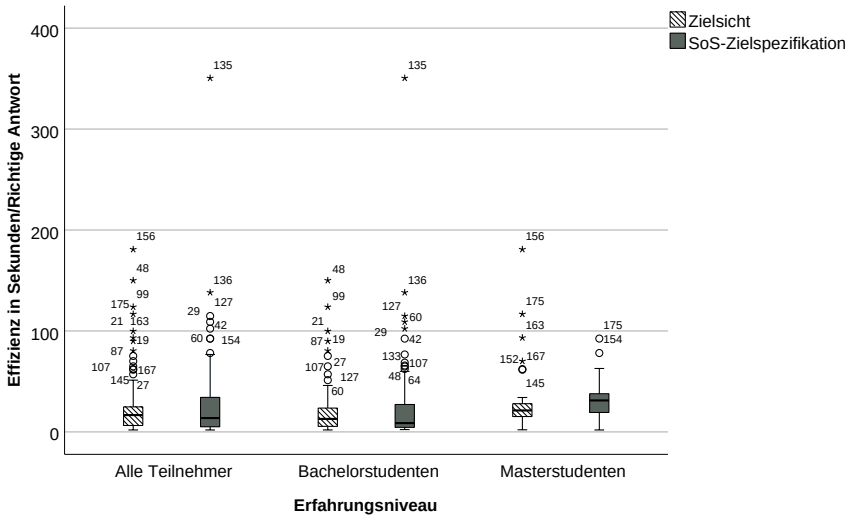


Abbildung 10.8: Effizienz der Zielsicht

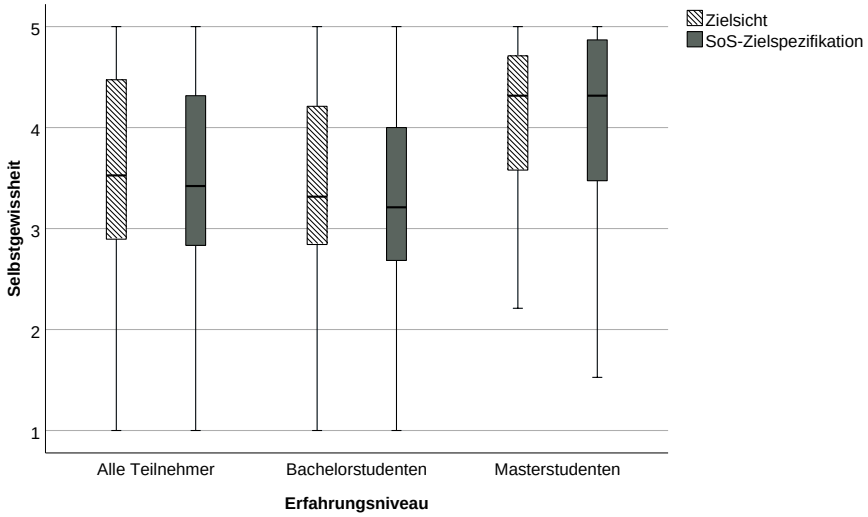


Abbildung 10.9: Selbstgewissheit bei der Zielsicht

SoS-Konfigurationen. Auch hier ist erkennbar, dass die Unterschiede zwischen Zielsicht und SoS-Zielspezifikation gering sind. Ein deutlicher Unterschied zeichnet sich hingegen zwischen Bachelor- und Masterstudenten ab.

10.4.1.4 Hypothesentests

Im folgenden Abschnitt werden die Ergebnisse der Hypothesentests für die Zielsicht vorgestellt. Es wurde jeweils der Haupteffekt des Darstellungsformats (d. h. Zielsicht oder SoS-Zielspezifikation) und des Erfahrungsniveaus (d. h. Bachelor- oder Masterstudenten) untersucht. Des weiteren wurde die Interaktion zwischen den beiden Faktoren Darstellungsformat und Erfahrungsniveau untersucht, um festzustellen, ob die Darstellungsformate bei den beiden Teilnehmergruppen zu unterschiedlichen Effekten auf die abhängige Variable führen.

Effektivität

Eine gemischte Varianzanalyse zeigt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen statistisch signifikanten Effekt der Darstellungsform auf die Effektivität $F(1, 179) = 39, 43, p < 0, 001$. Die Alternativhypothese $H_{1.1}a$ wird angenommen. Das bedeutet, die Zielsicht unterstützt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.

Außerdem ist ein signifikanter Haupteffekt des Erfahrungsniveaus aufgetreten, sodass sich bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen ein statistisch signifikanter Unterschied zwischen den Erfahrungsniveaus bzgl. der Effektivität ergeben hat, $F(1, 179) = 33, 16, p < 0, 001$. Die Alternativhypothese $H^B_{1.1}b$ wird angenommen. Das bedeutet, Masterstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Bachelorstudenten.

Bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen zeigte sich eine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Effektivität, $F(1, 179) = 12, 77, p < 0, 001$. Die Alternativhypothese $H^I_{1.1}a$ wird angenommen. Zugleich gibt es einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews. Das bedeutet, dass sich der Effekt der Zielsicht auf die Effektivität des Reviews signifikant zwischen Bachelor- und Masterstudenten unterscheidet.

Effizienz

Eine gemischte Varianzanalyse zeigt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen statistisch signifikanten Effekt, der von der Darstellungsform auf die Effizienz $F(1, 179) = 0, 19, p = 0, 66$ ausgeübt wird. Die Nullhypothese $H_{1.2}0$ kann nicht verworfen werden. Das bedeutet, das Darstellungsformat hat bei Fragestellungen bzgl. der

Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Effizienz des Reviews.

Allerdings ist ein signifikanter Haupteffekt des Erfahrungsniveaus aufgetreten, was bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen statistisch signifikanten Unterschied zwischen den Erfahrungsniveaus bzgl. der Effizienz zeigt, $F(1, 179) = 141, 13, p = 0, 03$. Die Alternativhypothese $H^B_{1.2}a$ wird angenommen. Das bedeutet, Bachelorstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Masterstudenten.

Es gab bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Effizienz, $F(1, 179) = 0, 28, p = 0, 59$. Die Nullhypothese $H^I_{1.2}0$ kann nicht verworfen werden. Zudem trat kein signifikanter Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews ein. Das bedeutet, dass sich der Effekt der Zielsicht auf die Effizienz des Reviews zwischen Bachelor- und Masterstudenten nicht signifikant unterscheidet.

Selbstgewissheit

Eine gemischte Varianzanalyse zeigt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen statistisch signifikanten Effekt, der von der Darstellungsform auf die Selbstgewissheit $F(1, 179) = 1, 18, p = 0, 28$ ausgeht. Die Nullhypothese $H_{1.3}0$ kann nicht verworfen werden. Das bedeutet, das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.

Allerdings ergab sich ein signifikanter Haupteffekt des Erfahrungsniveaus, was bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen statistisch signifikanten Unterschied zwischen den Erfahrungsniveaus bzgl. Selbstgewissheit zeigt, $F(1, 179) = 16, 03, p < 0, 001$. Die Alternativhypothese $H^B_{1.3}b$ wird angenommen. Das bedeutet, Masterstudenten haben bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen eine höhere Selbstgewissheit als Bachelorstudenten.

Bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen existiert keine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Selbstgewissheit ab, $F(1, 179) = 0, 39, p = 0, 53$. Die Nullhypothese $H^I_{1.3}0$ kann nicht verworfen werden. Bei diesen Fragestellungen ergab sich zudem kein signifikanter Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers. Das bedeutet, dass sich der Effekt der Zielsicht auf die Selbstsicherheit des Reviewers nicht signifikant zwischen Bachelor- und Masterstudenten unterscheidet.

Zusammenfassung der Ergebnisse der Hypothesentests

Tabelle 10.8 enthält eine Übersicht über die angenommenen Alternativ- und nicht verworfenen Nullhypothesen für die Zielsicht.

10.4.2 SoS-Konfigurationssicht

In diesem Abschnitt werden die Hypothesen, das Verfahren zur Auswertung sowie die Ergebnisse des Vergleichs der SoS-Konfigurationssicht mit der SoS-Zielspezifikation behandelt.

10.4.2.1 Hypothesen für die SoS-Konfigurationssicht

Für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit werden jeweils eine Nullhypothese für die beiden Haupteffekte (Darstellungsformat und Erfahrungsniveau) und die beiden gerichteten Alternativhypothesen definiert. Außerdem wird für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit eine Nullhypothese für den Interaktionseffekt zwischen Darstellungsformat und Erfahrungsniveau sowie eine ungerichtete Alternativhypothese für das Vorhandensein einer Interaktion festgelegt. Es ergeben sich ergänzend die folgenden Hypothesen:

Hypothesen zur ***Effektivität***:

- $H_{2.1}0$ Das Darstellungsformat hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Effektivität des Reviews.
- $H_{2.1}a$ Die SoS-Konfigurationssicht unterstützt einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
- $H_{2.1}b$ Die SoS-Zielspezifikation unterstützt einen signifikant effektiveren Review bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können.
- $H^B_{2.1}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Effektivität des Reviews.
- $H^B_{2.1}a$ Bachelorstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Masterstudenten.
- $H^B_{2.1}b$ Masterstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Bachelorstudenten.

Tabelle 10.8: Übersicht über die Ergebnisse der Hypothesentests für die Zielsicht

Hypothese	p	Bedeutung
$H_{1.1}a$	<0,001	Die Zielsicht unterstützt einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
$H^B_{1.1}b$	<0,001	Masterstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effektivere Reviews durch als Bachelorstudenten.
$H^I_{1.1}a$	<0,001	Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.
$H_{1.2}0$	0,66	Das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Effizienz des Reviews.
$H^B_{1.2}a$	0,03	Bachelorstudenten führen bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen signifikant effizienter Reviews durch als Masterstudenten.
$H^I_{1.2}0$	0,59	Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.
$H_{1.3}0$	0,28	Das Darstellungsformat hat bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.
$H^B_{1.3}b$	<0,001	Masterstudenten haben bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen eine höhere Selbstgewissheit als Bachelorstudenten.
$H^I_{1.3}0$	0,53	Es gibt bei Fragestellungen bzgl. der Zielerfüllung ausgewählter SoS-Konfigurationen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.

- $H^I_{2.1}0$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.
- $H^I_{2.1}a$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.

Hypothesen zur ***Effizienz***:

- $H_{2.2}0$ Das Darstellungsformat hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Effizienz des Reviews.
- $H_{2.2}a$ Die SoS-Konfigurationssicht unterstützt einen signifikant effizienteren Review im Vergleich zur SoS-Zielspezifikation.
- $H_{2.2}b$ Die SoS-Zielspezifikation unterstützt im Vergleich zur SoS-Konfigurationssicht einen signifikant effizienteren Review bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können.
- $H^B_{2.2}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Effizienz des Reviews.
- $H^B_{2.2}a$ Bachelorstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Masterstudenten.
- $H^B_{2.2}b$ Masterstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effizientere Reviews durch als Bachelorstudenten.
- $H^I_{2.2}0$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.
- $H^I_{2.2}a$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.

Hypothesen zur ***Selbstgewissheit***:

- $H_{2.3}0$ Das Darstellungsformat hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.

- $H_{2.3}a$ Die SoS-Konfigurationssicht unterstützt im Vergleich zur SoS-Zielspezifikation eine signifikant höhere Selbstgewissheit des Reviewers.
- $H_{2.3}b$ Die SoS-Zielspezifikation unterstützt im Vergleich zur SoS-Konfigurationssicht eine höhere Selbstgewissheit des Reviewers bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können.
- $H^B_{2.3}0$ Das Erfahrungsniveau des Reviewers hat bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Einfluss auf die Selbstgewissheit des Reviewers.
- $H^B_{2.3}a$ Bachelorstudenten haben bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, eine höhere Selbstgewissheit als Masterstudenten.
- $H^B_{2.3}b$ Masterstudenten haben bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, eine höhere Selbstgewissheit als Bachelorstudenten.
- $H^I_{2.3}0$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.
- $H^I_{2.3}a$ Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.

10.4.2.2 Auswertung

Zur Prüfung der Hypothesen zu Effektivität, Effizienz und Selbstgewissheit wird eine gemischte Varianzanalyse (mixed ANOVA) durchgeführt. Dadurch wird sowohl der Within-Subject-Faktor Darstellungsformat als auch der Between-Subject-Faktor Erfahrungsniveau untersucht.

10.4.2.3 Deskriptive Statistiken

In diesem Abschnitt werden die Ergebnisse für die abhängigen Variablen Effektivität, Effizienz und Selbstgewissheit für die SoS-Konfigurationssicht dargestellt. Die SoS-Konfigurationssicht zeigt, für welche SoS-Konfigurationen die ausgewählten intentionalen Elemente eines Zielmodells als erfüllbar spezifiziert sind. Um den Nutzen der SoS-Konfigurationssicht zu untersuchen, wurde die Nutzung der SoS-Konfigurationssicht mit der Nutzung der SoS-Zielspezifikation verglichen.

Effektivität

Tabelle 10.9 beinhaltet die deskriptive Statistik für die Effektivität der SoS-Konfigurationssicht. Die Daten zeigen, dass bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, die Nutzung der SoS-Konfigurationssicht ($M = 62,32\%, SD = 21,22\%$) eine höhere Effektivität aufweist als die Nutzung der SoS-Zielspezifikation ($M = 57,85\%, SD = 19,59\%$). Außerdem wird deutlich, dass Masterstudenten sowohl bei der Nutzung der SoS-Konfigurationssicht als auch bei der Nutzung der SoS-Zielspezifikation effektiver sind als Bachelorstudenten bei der Nutzung des jeweiligen Darstellungsformats. Abbildung 10.10 illustriert die Verteilung für die abhängige Variable Effektivität der SoS-Konfigurationssicht und der SoS-Zielspezifikation bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können. Es ist erkennbar, dass bei Masterstudenten der Unterschied in der Effektivität zwischen der SoS-Konfigurationssicht und der SoS-Zielspezifikation stärker ausgeprägt ist als bei Bachelorstudenten.

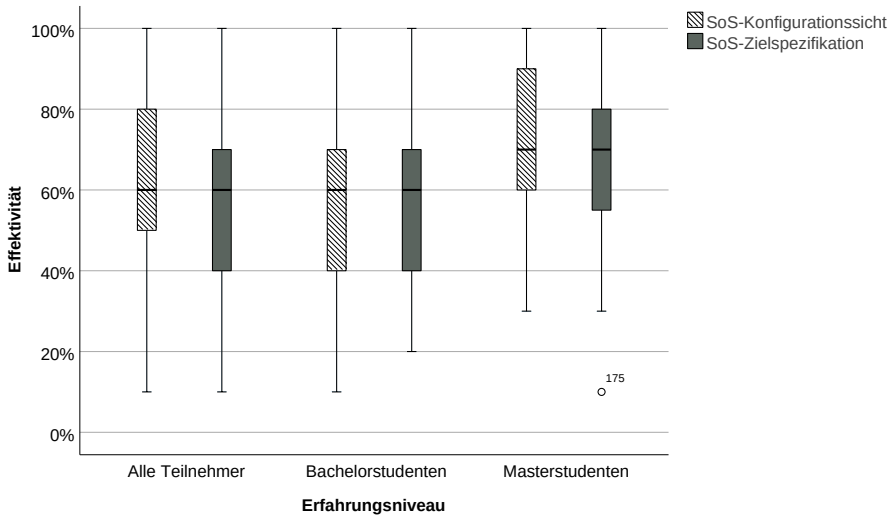


Abbildung 10.10: Effektivität der SoS-Konfigurationssicht

Effizienz

Tabelle 10.10 beinhaltet die deskriptive Statistik für die Effizienz der SoS-Konfigurationssicht. Die Daten zeigen, dass die Nutzung der SoS-Konfigurationssicht ($M = 49,74, SD = 57,95$) im arithmetischen Mittel bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, effizienter ist als die Nutzung der SoS-Zielspezifikation ($M = 67,98, SD = 199,34$). Bei Betrachtung der Mediane zeichnet sich jedoch ein gegenteiliges Bild ab. Auffallend ist auch hier vor allem die höhere Effizienz der Bachelorstudenten

Tabelle 10.9: Deskriptive Statistik - Effektivität der SoS-Konfigurationssicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
SoS-Konfigurationssicht	Alle Teilnehmer	181	62,32%	59,21%	65,43%	60,00%	21,22%	10,00%	100,00%	30,00%
	Bachelorstudenten	142	59,23%	55,84%	62,61%	60,00%	20,39%	10,00%	100,00%	30,00%
	Masterstudenten	39	73,59%	66,92%	80,26%	70,00%	20,58%	30,00%	100,00%	30,00%
SoS-Ziel-	Alle Teilnehmer	181	57,85%	54,97%	60,72%	60,00%	19,59%	10,00%	100,00%	30,00%
spezifikation	Bachelorstudenten	142	56,06%	52,95%	59,16%	60,00%	18,72%	20,00%	100,00%	30,00%
	Masterstudenten	39	64,36%	57,39%	71,33%	70,00%	21,50%	10,00%	100,00%	30,00%

Tabelle 10.10: Deskritpive Statistik - Effizienz der SoS-Konfigurationssicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
SoS-Konfigurationssicht	Alle Teilnehmer	181	49,74	41,24	58,24	35,14	57,95	2,71	357,00	55,56
	Bachelorstudenten	142	44,42	35,02	53,82	24,75	56,68	2,71	357,00	49,00
	Masterstudenten	39	69,10	49,94	88,26	51,00	59,10	3,14	338,33	38,15
SoS-Ziel-	Alle Teilnehmer	181	67,98	38,75	97,22	18,50	199,34	2,29	1867,00	53,74
spezifikation	Bachelorstudenten	142	48,20	24,47	71,94	13,00	143,05	2,29	1617,20	48,54
	Masterstudenten	39	139,99	34,70	245,29	48,29	324,82	3,14	1867,00	50,56

Tabelle 10.11: Deskriptive Statistik - Selbstgewissheit bei der SoS-Konfigurationssicht

				95% Konfidenzintervall des Mittelwerts			Standard-			Interquartil-
		N	Mittelwert	Untergrenze	Obergrenze	Median	abweichung	Minimum	Maximum	bereich
SoS-Konfigurationssicht	Alle Teilnehmer	181	3,45	3,30	3,60	3,40	1,02	1,00	5,00	1,53
	Bachelorstudenten	142	3,35	3,18	3,51	3,32	1,00	1,00	5,00	1,30
	Masterstudenten	39	3,84	3,50	4,18	4,20	1,04	1,40	5,00	1,90
SoS-Ziel-	Alle Teilnehmer	181	3,29	3,13	3,45	3,20	1,08	1,00	5,00	1,30
spezifikation	Bachelorstudenten	142	3,19	3,01	3,36	3,20	1,07	1,00	5,00	1,13
	Masterstudenten	39	3,67	3,33	4,01	3,60	1,05	1,50	5,00	1,60

($Mdn = 24,75$ für die SoS-Konfigurationssicht und $Mdn = 13,00$ für die SoS-Zielspezifikation) verglichen mit den Masterstudenten ($Mdn = 51,00$ für die SoS-Konfigurationssicht und $Mdn = 48,29$ für die SoS-Zielspezifikation).

Abbildung 10.11 illustriert die Verteilung für die Effizienz der SoS-Konfigurationssicht und der SoS-Zielspezifikation bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können. Insbesondere sind die vielen Ausreißer mit geringer Effizienz auffällig. Dies erklärt auch die Unterschiede zwischen den arithmetischen Mitteln und den Medianen.

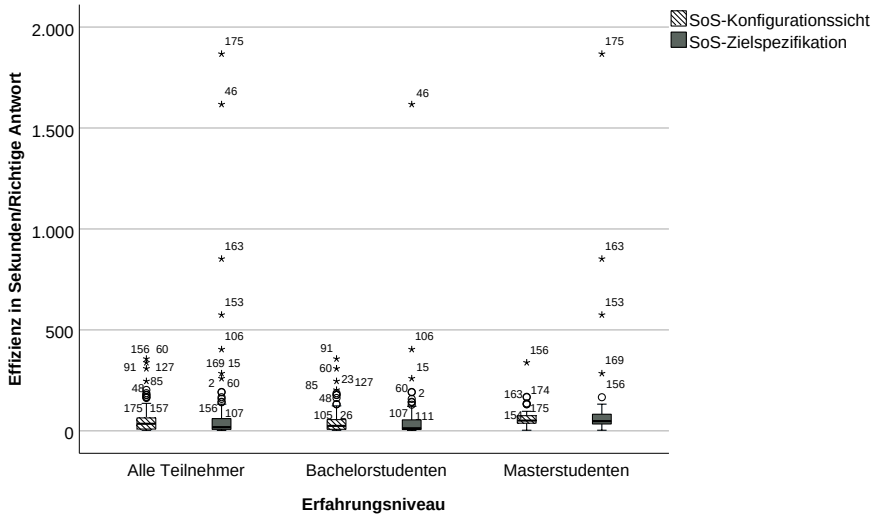


Abbildung 10.11: Effizienz der SoS-Konfigurationssicht

Selbstgewissheit

Tabelle 10.11 kann die deskriptive Statistik für die Selbstgewissheit bei der Nutzung der SoS-Konfigurationssicht entnommen werden. Gemäß den Daten wird bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, die Selbstgewissheit bei der Nutzung der SoS-Konfigurationssicht ($M = 3,45, SD = 1,02$) durchschnittlich höher eingeschätzt als bei der Nutzung der SoS-Zielspezifikation ($M = 3,29, SD = 1,08$).

Abbildung 10.12 illustriert die Verteilung für die abhängige Variable Selbstgewissheit bei der Nutzung der SoS-Konfigurationssicht und der SoS-Zielspezifikation. Wie zu sehen ist, besteht vor allem ein Unterschied in der Selbstgewissheit zwischen Bachelor- und Masterstudenten. Der Unterschied zwischen der Selbstgewissheit bei der Nutzung von SoS-Konfigurationssicht und bei der Nutzung der SoS-Zielspezifikation ist deutlich geringer ausgeprägt.

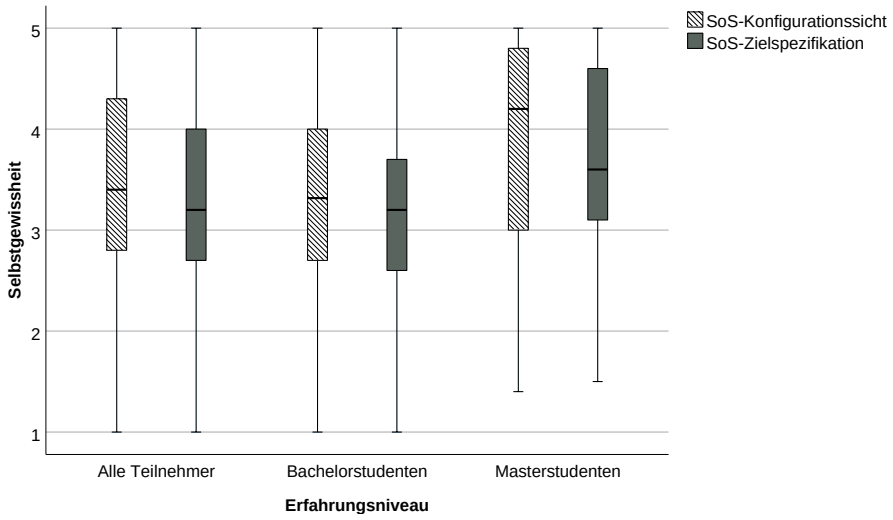


Abbildung 10.12: Selbstgewissheit bei der SoS-Konfigurationssicht

10.4.2.4 Hypothesentests

Nachfolgend werden die Ergebnisse der Hypothesentests für die SoS-Konfigurationssicht vorgestellt. Es wurde jeweils der Haupteffekt des Darstellungsformats (d. h. SoS-Konfigurationssicht oder SoS-Zielspezifikation) und des Erfahrungsniveaus (d. h. Bachelor- oder Masterstudenten) untersucht. Des Weiteren wurde die Interaktion zwischen den beiden Faktoren Darstellungsformat und Erfahrungsniveau betrachtet, um festzustellen, ob die Darstellungsformate bei den beiden Teilnehmergruppen zu unterschiedlichen Effekten auf die abhängige Variable führen.

Effektivität

Eine gemischte Varianzanalyse zeigt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Effekt der Darstellungsform auf die Effektivität $F(1, 179) = 11, 3, p = 0, 001$. Die Alternativhypothese $H_{2.1}a$ wird angenommen. Das bedeutet, die SoS-Konfigurationssicht unterstützt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.

Außerdem hat sich ein signifikanter Haupteffekt des Erfahrungsniveaus ergeben, was bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Unterschied zwischen den Erfahrungsniveaus bzgl. Effektivität zeigt, $F(1, 179) = 13, 44, p < 0, 001$. Die Alternativhypothese $H^B_{2.1}b$ wird angenommen. Das bedeutet, Masterstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die

ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Bachelorstudenten.

Bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, stellte sich keine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Effektivität heraus, $F(1, 179) = 2, 72, p = 0, 10$. Die Nullhypothese $H^I_{2.1}0$ kann nicht verworfen werden. Es gibt bei diesen Fragestellungen zudem keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews. Das bedeutet, dass sich der Effekt der SoS-Konfigurationssicht auf die Effektivität des Reviews nicht signifikant zwischen Bachelor- und Masterstudenten unterscheidet.

Effizienz

Eine gemischte Varianzanalyse zeigt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Effekt der Darstellungsform auf die Effizienz $F(1, 179) = 4, 71, p = 0, 03$. Die Alternativhypothese $H_{2.2}a$ wird angenommen. Das bedeutet, die SoS-Konfigurationssicht unterstützt einen signifikant effizienteren Review im Vergleich zur SoS-Zielspezifikation.

Außerdem ist ein signifikanter Haupteffekt des Erfahrungsniveaus aufgetreten, was bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Unterschied zwischen den Erfahrungsniveaus bzgl. Effizienz nahelegt, $F(1, 179) = 8, 77, p = 0, 003$. Die Alternativhypothese $H^B_{2.2}a$ wird angenommen. Demzufolge führen Bachelorstudenten bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Masterstudenten.

Bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, gibt es keine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Effizienz, $F(1, 179) = 3, 80, p = 0, 53$. Die Nullhypothese $H^I_{2.2}0$ kann nicht verworfen werden. Diese Fragestellungen weisen keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews auf. Das bedeutet, dass sich der Effekt der SoS-Konfigurationssicht auf die Effizienz des Reviews nicht signifikant zwischen Bachelor- und Masterstudenten unterscheidet.

Selbstgewissheit

Eine gemischte Varianzanalyse zeigt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Effekt der Darstellungsform auf die Selbstgewissheit $F(1, 179) = 5, 71, p = 0, 02$. Die Alternativhypothese $H_{2.3}a$

wird angenommen. Das bedeutet, die SoS-Konfigurationssicht unterstützt eine signifikant höhere Selbstgewissheit des Reviewers im Vergleich zur SoS-Zielspezifikation.

Außerdem ist ein signifikanter Haupteffekt des Erfahrungsniveaus aufgetreten, was bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, einen statistisch signifikanten Unterschied zwischen den Erfahrungsniveaus bzgl. Selbstgewissheit nachweist, $F(1, 179) = 16,03, p < 0,001$. Die Alternativhypothese $H^B_{2.3}b$ wird angenommen. Das bedeutet, Masterstudenten haben bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, eine höhere Selbstgewissheit als Bachelorstudenten.

Es gab bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keine statistisch signifikante Interaktion zwischen der Darstellungsform und dem Erfahrungsniveau in Bezug auf die Selbstgewissheit, $F(1, 179) = 0,01, p = 0,93$. Die Nullhypothese $H^I_{2.3}0$ kann nicht verworfen werden. Für diese Art Fragestellungen existiert kein signifikanter Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers. Demzufolge unterscheidet sich der Effekt der SoS-Konfigurationssicht auf die Selbstsicherheit des Reviewers nicht signifikant zwischen Bachelor- und Masterstudenten.

Zusammenfassung der Ergebnisse der Hypothesentests für die Zielsicht

Tabelle 10.12 gibt eine Übersicht über die angenommenen Alternativ- und nicht verworfenen Nullhypothesen für die Zielsicht.

Tabelle 10.12: Übersicht über die Ergebnisse der Hypothesentests für die SoS-Konfigurationssicht

Hypothese	p	Bedeutung
$H_{2.1}a$	0,001	Die SoS-Konfigurationssicht unterstützt einen signifikant effektiveren Review im Vergleich zur SoS-Zielspezifikation.
$H^B_{2.1}b$	<0,001	Masterstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effektivere Reviews durch als Bachelorstudenten.
$H^I_{2.1}0$	0,10	Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effektivität des Reviews.
$H_{2.2}a$	0,03	Die SoS-Konfigurationssicht unterstützt einen signifikant effizienteren Review im Vergleich zur SoS-Zielspezifikation.
$H^B_{2.2}a$	0,003	Bachelorstudenten führen bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, signifikant effizientere Reviews durch als Masterstudenten.
$H^I_{2.2}0$	0,53	Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Effizienz des Reviews.
$H_{2.3}a$	0,02	Die SoS-Konfigurationssicht unterstützt eine signifikant höhere Selbstgewissheit des Reviewers im Vergleich zur SoS-Zielspezifikation.
$H^B_{2.3}b$	<0,001	Masterstudenten haben bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, eine höhere Selbstgewissheit als Bachelorstudenten.
$H^I_{2.3}0$	0,93	Es gibt bei Fragestellungen, ob bestimmte SoS-Konfigurationen die ausgewählten Ziele erfüllen können, keinen signifikanten Interaktionseffekt zwischen der Darstellungsform und dem Erfahrungsniveau des Teilnehmers bzgl. der Selbstgewissheit des Reviewers.

10.5 Diskussion der Experimentergebnisse

Dieser Abschnitt diskutiert die in den vorherigen Abschnitten vorgestellten Experimentergebnisse. Zunächst werden in 10.5.1 die Ergebnisse des Experiments noch einmal überblicksartig zusammengefasst. Abschnitt 10.5.2 adressiert die Validitätsgefährdungen, die möglicherweise Einfluss auf die Gültigkeit der Experimentergebnisse nehmen können, und beschreibt die Maßnahmen, die getroffen wurden, um Validitätsgefährdungen so weit wie möglich zu reduzieren. Abschließend werden die Experimentergebnisse in 10.5.3 unter Berücksichtigung der Validitätsgefährdungen mit Blick auf die Forschungsfragestellungen interpretiert.

10.5.1 Zusammenfassung der Ergebnisse

Das Experiment hat gezeigt, dass:

- der Review von Sichten signifikant effektiver ist als der Review von SoS-Zielspezifikationen. ($p < 0{,}001$)
- es keinen signifikanten Unterschied hinsichtlich der Effizienz zwischen dem Review von Sichten und SoS-Zielspezifikationen gibt. ($p = 0{,}15$)
- die Nutzung von Sichten beim Review zu einer signifikant höheren Selbstgewissheit des Reviewers im Vergleich zur Nutzung von SoS-Zielspezifikationen führt. ($p = 0{,}01$)
- die Sichten von den Teilnehmern subjektiv als unterstützender bewertet werden. ($p = 0{,}01$)

Durch die erweiterte Auswertung wurde deutlich, dass vor allem die Nutzung der SoS-Konfigurationssicht Vorteile bietet.

- Die Nutzung einer SoS-Konfigurationssicht ist beim Review signifikant effektiver als die Nutzung der SoS-Zielspezifikation. ($p = 0{,}001$)
- Die Nutzung einer SoS-Konfigurationssicht ist beim Review signifikant effizienter als die Nutzung der SoS-Zielspezifikation. ($p = 0{,}03$)
- Die Nutzung einer SoS-Konfigurationssicht führt beim Review zu einer signifikant höheren Selbstgewissheit. ($p = 0{,}02$)

Für die Nutzung der Zielsicht haben sich im Experiment vor allem Vorteile bzgl. der Effektivität gezeigt.

- Die Nutzung einer Zielsicht ist beim Review signifikant effektiver als die Nutzung der SoS-Zielspezifikation. ($p < 0{,}001$)
- Es gibt keinen signifikanten Unterschied hinsichtlich der Effizienz zwischen dem Review einer Zielsicht und der SoS-Zielspezifikation. ($p = 0{,}66$)

- Es gibt keinen signifikanten Unterschied hinsichtlich der Selbstgewissheit des Reviewers zwischen der Nutzung einer Zielsicht und der SoS-Zielspezifikation. (p = 0,28)

Das Experiment hat außerdem gezeigt, dass Masterstudenten effektiver und selbstgewisser agieren als Bachelorstudenten. Überraschenderweise lag jedoch die Effizienz bei Bachelorstudenten höher als bei Masterstudenten. Ein signifikanter Interaktionseffekt zeichnete sich ausschließlich bei der Effektivität der Zielsicht ab. Das bedeutet, dass erfahrene Reviewer einen besonders starken Nutzen bzgl. der Effektivität bei Nutzung der Zielsicht haben. Diese Ergebnisse legen nahe, dass vor allem erfahrenere Reviewer durch die Sichtenbildung unterstützt werden.

10.5.2 Diskussion der Validitätsgefährdungen

Dieser Abschnitt beschreibt die existierenden Validitätsgefährdungen für diese Art von Experiment (vgl. (Campbell und Stanley 1963; Wohlin et al. 2012)) und wie diese adressiert wurden. In diesem Zusammenhang wird auf die interne Validität, die Konstruktvalidität, die Schlussfolgerungsvalidität und die externe Validität eingegangen. Die interne Validität betrifft die Eindeutigkeit der Untersuchungsergebnisse. Bei der Konstruktvalidität geht es darum, ob sich aus den Ergebnissen des Experiments Rückschlüsse auf die Fragestellung ableiten lassen. Im Rahmen der Schlussfolgerungsvalidität sollen richtige Schlüsse über die Zusammenhänge von Intervention und Ergebnis gezogen werden und die externe Validität betrifft die Generalisierbarkeit der Untersuchungsergebnisse.

10.5.2.1 Interne Validität

Typische Gefährdungen der internen Validität entstehen durch die Geschichte, d. h. die Ereignisse, die zwischen den Messungen stattfinden und diese beeinflussen, sowie die Reifung, d. h. das Älterwerden der Teilnehmer zwischen Messungen, oder deren Ausfall, d. h. der Wegfall von Teilnehmern zwischen Messungen. Um diese Gefährdungen zu adressieren, wurden die Daten für die Sichten und die SoS-Zielspezifikation unmittelbar hintereinander erhoben. Dadurch, dass jeder Teilnehmer sowohl die Sichten als auch die SoS-Zielspezifikation gereviewt hat, besteht die Gefahr, dass Lerneffekte, die sich aus den ersten Fragen ergeben, die späteren Antworten beeinflussen. Um diese Gefahr auszuschließen, wurde die Reihenfolge der Modelle randomisiert. Eine weitere Gefährdung resultiert daraus, dass Teilnehmer erahnen, was das gewünschte Ergebnis ist, und ihr Verhalten daran anpassen. Um dies zu verhindern, haben die Teilnehmer vorab keine detaillierten Informationen über das Experiment erhalten, sondern wurden nur informiert, dass es um den Review von Modellen geht. Die Teilnehmer erhielten keine Belohnungen wie Bonuspunkte, um zu verhindern, dass die Ergebnisse durch besondere Anstrengungen seitens der Teilnehmer verzerrt werden. Bei den Teilnehmern handelt es sich jedoch um eine willkürliche Stichprobe.

10.5.2.2 Konstruktvalidität

Der Experimentaufbau ist angelehnt an ähnliche Untersuchungen (Daun et al. 2021). Dies erlaubt nicht nur die Vergleichbarkeit, sondern verhindert auch den Einfluss von Fehlern im Experimentaufbau auf Messwerte. Es wurden ausschließlich quantitative Daten erhoben, was die Gefahr der Fehlinterpretation reduziert. Probleme bestehen jedoch bei der Zeitmessung und damit bei der daraus ermittelten Effizienz. Der Fragebogen erlaubt lediglich eine Zeitmessung pro Modell, daher übt der Anteil der falschen Antworten einen hohen Einfluss auf die ermittelte Effizienz aus. Außerdem hat sich anhand der Zeitmessung gezeigt, dass vor allem unter den Bachelorstudenten einige Teilnehmer das Experiment nicht ernsthaft durchgeführt haben. Dies könnte eine mögliche Erklärung für die durchschnittlich höhere Effizienz von Bachelorstudenten sein, da bei einem zufälligen Raten mit einer Effektivität von durchschnittlich 50% zu rechnen ist. Jedoch war es nicht möglich, objektiv zu bestimmen, ab welchem Zeitrahmen von einer ernsthaften Teilnahme ausgegangen werden konnte. Da aufgrund des Within-Subject-Designs nicht die Werte der Teilnehmer untereinander, sondern die Werte für jeden Teilnehmer mit seinen eigenen verglichen werden, besteht nicht die Gefahr einer fälschlichen Erkennung von Unterschieden, sondern es kann auch vorkommen, dass Unterschiede nicht erkannt werden. Außerdem übt dieses Verhalten Einfluss auf die Lage- und Streuparameter wie Mittelwert, Median und Varianz aus. Ebenfalls traten bei der Zeitmessung vereinzelt Ausreißer mit einer sehr langen Durchführungsdauer auf. Davon war vor allem der Review der SoS-Zielspezifikation betroffen, bei der aufgrund des größeren Umfangs eine längere Review-Zeit zu erwarten war. Eine Bereinigung dieser Datensätze hätte zu einer Verzerrung der Werte geführt. Insbesondere war zu erkennen, dass in vielen dieser Fälle auch die Dauer für den Review der Sichten deutlich über den Durchschnittswerten lag. So ist zumindest in manchen Fällen davon auszugehen, dass der Teilnehmer sehr gewissenhaft an dem Experiment teilgenommen hat. Aus diesen Gründen wurden auch diese Datensätze mit ausgewertet. Das kontrollierte Experiment hat gezeigt, dass die Sichten den Review von SoS-Zielspezifikationen unterstützen. Allerdings beinhaltet das Experiment weder die Auswahl der generierten Sichten noch die Korrektur möglicher Fehler. Das bedeutet, dass eine andere Auswahl von Sichten möglicherweise zu anderen Ergebnissen geführt hätte.

10.5.2.3 Schlussfolgerungsvalidität

Zugunsten einer hohen Schlussfolgerungsvalidität wurde auf eine hohe Teilnehmerzahl geachtet. Dies reduziert die Gefahr, dass existierende Unterschiede aufgrund von zu wenigen Teilnehmern nicht erkannt werden. Das Experiment hat statistisch signifikante Unterschiede zwischen der Nutzung der Sichten und der SoS-Zielspezifikation aufgezeigt. Daher ist davon auszugehen, dass Effekte bei den nicht statistisch signifikanten abhängigen Variablen erkannt worden wären, wenn diese vorhanden wären. Um sicherzustellen, dass das Experimentmaterial von den Teilnehmern verstanden wird, wurde ein Vortest durchgeführt.

10.5.2.4 Externe Validität

Die externe Validität beschäftigt sich damit, ob die Ergebnisse generalisierbar sind. Als Teilnehmer wurden Bachelor- und Masterstudenten ausgewählt, um den Effekt von unterschiedlichen Erfahrungsstufen genauer zu untersuchen. Frühere Experimente haben gezeigt, dass Studierende in gewissen Situationen generalisierbar sind und teilweise sogar Vorteile gegenüber der Nutzung von Berufstätigen bieten (Daun et al. 2016). Der Einsatz von Industriepartnern hätte eine deutlich geringere Teilnehmerzahl zur Folge, wodurch sich wiederum die Gefährdung der Schlussfolgerungsvalidität erhöht hätte. Das Experimentmaterial wurde in Zusammenarbeit mit Industrieexperten erstellt, um Realitätstreue zu garantieren.

10.5.3 Interpretation und Generalisierbarkeit der Ergebnisse

Das Experiment hat gezeigt, dass die Nutzung von Sichten Vorteile bei der Validierung von SoS-Zielspezifikationen bietet. Besonders hervorzuheben sind die

- erhöhte Effektivität bei der Nutzung der Sichten
- erhöhte subjektive Unterstützungsfähigkeit bei der Nutzung der Sichten
- erhöhte Selbstgewissheit bei der Nutzung der SoS-Konfigurationssicht

Die Steigerung der Effektivität, deutet darauf hin, dass es Reviewern bei der Nutzung der Sichten leichter fällt, Defekte zu erkennen. Allerdings steht diese gesteigerte Effektivität möglicherweise einer reduzierten Effizienz gegenüber. Da die Sichten weniger Informationen enthalten als die SoS-Zielspezifikation ist es erforderlich, diverse Sichten zu reviewen. Das Experiment konnte nicht zeigen, dass die Effizienz bei dem Review einer Sicht höher ist als bei dem Review der SoS-Zielspezifikation bzgl. der gleichen Aspekte. Daher ist zu vermuten, dass die Nutzung von Sichten bei der Validierung den Aufwand der Validierung erhöht. Wie in Abschnitt 10.5.2 diskutiert wurde, unterliegen die Effizienzwerte einem stark Einfluss von Ausreißern, wodurch eine zuverlässige Aussage erschwert wird. Bzgl. der Generalisierbarkeit hat sich gezeigt, dass vor allem bei Masterstudenten die Vorteile der Nutzung der Sichten zum Tragen kommen. Dies deutet darauf hin, dass auch Berufserfahrene in der industriellen Praxis von der Nutzung der Sichten profitieren können.

Insgesamt legen die Ergebnisse damit nahe, dass der vorgeschlagene Ansatz vorteilhaft ist. Somit kann die Evaluationsfragestellung E2 als erfüllt erachtet werden.

Teil IV

Zusammenfassung und Ausblick

11 Kapitel
Fazit und Ausblick

Diese Arbeit stellt einen Ansatz zur Unterstützung der Spezifikation und Validierung von Zielen für Systems-of-Systems vor. Hierbei wurde insbesondere die Vielzahl von unterschiedlichen SoS-Konfigurationen berücksichtigt. Durch die automatisierte Generierung von Zielsichten und SoS-Konfigurationssichten wird die Prüfung der SoS-Zielspezifikation erleichtert. Der Einsatz von Konsistenzprüfungsansätzen unterstützt die Prüfung durch die Identifizierung von automatisiert erkennbaren Fehlern.

Dieses Kapitel fasst die Ergebnisse der Arbeit abschließend zusammen. Abschnitt 11.1 greift die Beiträge auf, die diese Arbeit leistet. Abschnitt 11.2 diskutiert Einschränkungen, die sich für die Ergebnisse der Arbeit ergeben, und Abschnitt 11.3 stellt mögliche zukünftige Forschungsfragestellungen zur Reduktion der existierenden Einschränkungen vor.

11.1 Beiträge der Arbeit

Einer der Vorteile von Systems-of-Systems ist ihre Fähigkeit, sich zu rekonfigurieren. D. h., ein System-of-Systems kann Bestandteile austauschen. Dadurch ergeben sich für ein System-of-Systems unterschiedliche SoS-Konfigurationen. Diese SoS-Konfigurationen müssen bei der Entwicklung eines System-of-Systems berücksichtigt werden. Für die zielorientierte Entwicklung von Systems-of-Systems bedeutet dies, dass nicht alle Ziele für alle SoS-Konfigurationen gewünscht sind. Dementsprechend müssen in einer Zielspezifikation für Systems-of-Systems nicht nur deren Ziele spezifiziert werden, sondern es ist auch festzuhalten, für welche SoS-Konfigurationen ein Ziel erfüllt werden soll. Bei Systems-of-Systems mit vielen unterschiedlichen SoS-Konfigurationen mit unterschiedlichen Zielen führt dies zu umfangreichen und komplexen Zielspezifikationen. Neben dieser besonderen Herausforderung bei der Spezifikation von Zielen für Systems-of-Systems wird ebenfalls die Validierung der zahlreichen möglichen Spezifikationen erschwert. Dadurch steigt die Gefahr, dass Zielspezifikationen für Systems-of-Systems Fehler enthalten, was wiederum die Entwicklung des gewünschten System-of-Systems beeinträchtigt.

Eine Analyse des Standes der Wissenschaft hat gezeigt, dass es Ansätze aus der Softwareproduktlinienentwicklung gibt, die die Spezifikation von Anforderungen für variable Systeme unterstützen. Dabei werden die Anforderungen und die Variabilität des Systems in getrennten Modellen spezifiziert und diese Modelle zueinander relationiert. Jedoch ergab sich eine Lücke bzgl. der Validierung der oft sehr komplexen Spezifikationen, d. h. der Prüfung, ob die Spezifikation die Stakeholder-Wünsche widerspiegelt.

Vor diesem Hintergrund bestand das Ziel dieser Arbeit darin, die Validierung von Zielspezifikationen für Systems-of-Systems bestehend aus einem Basis-Zielmodell und einem SoS-Modell zu unterstützen. Ein besonderes Augenmerk lag dabei auf der Spezifikation der Zusammenhänge zwischen Zielen und SoS-Konfigurationen. Diese Arbeit stellt in diesem Kontext einen Ansatz vor, der es erlaubt, Ziele für Systems-of-Systems mit unterschiedlichen SoS-Konfigurationen zu spezifizieren, und die Validierung solcher Spezifikationen unterstützt. Diese Unterstützung wird durch die automatisierte Generierung von Sichten realisiert. Der Nutzer kann sich basierend auf einer gültigen von ihm getroffenen Auswahl von Elementen des Zielmodells die SoS-Konfigurationen anzeigen lassen, für die die ausgewählten Elemente spezifiziert sind. Außerdem kann der Nutzer sich basierend auf einer gültigen von ihm getroffenen Auswahl von SoS-Konfigurationen die Elemente des Zielmodells anzeigen lassen, die für die ausgewählten SoS-Konfigurationen spezifiziert sind. Diese Sichten verdeutlichen die Zusammenhänge zwischen den Zielen und den SoS-Konfigurationen und erleichtern es, Fehler in Zielspezifikationen für Systems-of-Systems zu identifizieren.

Der Ansatz wurde durch eine prototypische Implementierung und ein kontrolliertes Experiment evaluiert. Die prototypische Implementierung hat gezeigt, dass der Ansatz sich technisch umsetzen lässt und die Sichten in akzeptabler Zeit automatisiert generiert werden können. Darüber hinaus wurde im Rahmen eines kontrollierten Experiments nachgewiesen, dass die Nutzung der Sichten bei der Validierung Vorteile bietet, insbesondere bzgl. der Identifikation von Fehlern.

11.2 Kritische Würdigung

Der in dieser Arbeit vorgeschlagene Ansatz zur Unterstützung der Spezifikation und Validierung von Zielen für Systems-of-Systems wurde durch eine prototypische Implementierung und ein kontrolliertes Experiment evaluiert. Im Rahmen der Evaluation wurde nachgewiesen, dass der vorgeschlagene Ansatz anwendbar ist. Die prototypische Implementierung zeigt, dass die Sichten automatisiert erzeugt werden können, und durch das kontrollierte Experiment wurde deutlich, dass die Nutzung der Sichten einen Vorteil bei der Validierung bietet. Insbesondere wurde ermittelt, dass die Nutzung der Sichten die Effektivität steigern kann. Dennoch existieren Einschränkungen, die zu berücksichtigen sind.

1. Erhöhter Aufwand der Validierung der Sichten verglichen mit einer Spezifikation

Im Rahmen des Experiments hat sich erwiesen, dass die Nutzung der Sichten zu einer effektiveren Validierung führt. Dies bedeutet, dass die Nutzung von Sichten es dem Nutzer erleichtert, Fehler in Zielspezifikationen für Systems-of-Systems zu finden. Jedoch ergab sich bei der Effizienz ein durchwachsenes Bild. Für die Nutzung der SoS-Konfigurationssicht zeigte sich ein signifikanter Unterschied zugunsten der SoS-Konfigurationssicht im Vergleich mit der SoS-Zielspezifikation. In Bezug auf die Zielsicht ließ sich ein solcher Unterschied jedoch nicht entdecken. Im Hinblick auf die Effizienz muss außerdem berücksichtigt werden, dass jede Sicht nur einen Ausschnitt der Spezifikation wiedergibt und somit Fehler außerhalb dieses Ausschnitts nicht identifiziert werden können. Daraus resultiert die Notwendigkeit, mehrere Sichten zu erstellen, wodurch sich der Aufwand wiederum erhöht.

2. Keine Unterstützung für die Identifikation der Ursache von Fehlern in Sichten

Der Ansatz unterstützt zwar die Identifikation von Fehlern in Zielspezifikationen für Systems-of-Systems, allerdings bietet er keine Hilfestellung für deren Korrektur. Entdeckt der Nutzer einen Fehler in einer Sicht, kann die Ursache ein Fehler im Zielmodell, ein Fehler im SoS-Modell, ein Fehler in den X-Links oder sogar ein Fehler bei der Auswahl für die Sichtenerstellung sein. Eine Kombination von verschiedenen Fehlern an unterschiedlichen Stellen ist ebenfalls möglich. Dies führt dazu, dass es in manchen Fällen schwierig sein kann, die Korrekturen in der SoS-Zielspezifikation durchzuführen. Allerdings kann hier die Erstellung weiterer Sichten basierend auf einer restriktiveren Auswahl helfen, den Fehler weiter einzugrenzen.

3. Keine Garantie auf Fehlerfreiheit

Darüber hinaus besteht – wie bei allen Validierungstätigkeiten – das Problem, dass Fehlerfreiheit nicht garantiert werden kann. Da es sich um eine manuelle Tätigkeit handelt, bei der die Spezifikation gegen nicht dokumentiertes Wissen geprüft wird, kann der Ansatz keine Sicherheit bieten, dass die validierte Zielspezifikation für ein System-of-Systems keine Fehler mehr enthält, wenn der Nutzer in den Sichten keine Fehler identifizieren konnte. D. h., der Ansatz unterstützt das Aufdecken von Fehlern, kann aber nicht das Nichtvorhandensein von Fehlern nachweisen.

11.3 Ausblick

Der vorgestellte Ansatz adressiert die Herausforderung der Validierung von Zielspezifikationen für Systems-of-Systems. Er unterstützt den Nutzer, indem er es erlaubt, Sichten auf Ziel- und SoS-Modelle zu erstellen, die die Komplexität reduzieren und somit die Identifikation von Fehlern vereinfachen. Basierend auf den Ergebnissen dieser Arbeit ergeben sich neue Fragestellungen für zukünftige Forschungsarbeiten.

1. Optimierung der Generierung von zielführenden Sichten

Hier stellt sich vor allem die Frage, ob es eine Möglichkeit gibt zu ermitteln, welche Sichten erstellt werden sollten. Der Nutzer kann sehr spezifische Sichten erstellen, bspw. basierend auf

nur einem ausgewählten Ziel oder nur einer ausgewählten SoS-Konfiguration. Eine Sicht erlaubt aber nur die Validierung eines vergleichsweise kleinen Ausschnitts der Spezifikation, was wiederum dazu führt, dass viele Sichten erstellt und überprüft werden müssen, um die gesamte SoS-Zielspezifikation zu validieren. Allerdings können basierend auf vielen ausgewählten Zielen oder SoS-Konfigurationen auch sehr grobe Sichten generiert werden. Diese decken zwar größere Ausschnitte der Spezifikation ab, können aber komplexer als spezifischere Sichten und dadurch schwieriger zu prüfen sein. Für die Nutzung des Ansatzes wäre es hilfreich, Hinweise zu haben, wie ein Kompromiss gefunden werden kann, der Aufwand und Nutzen gegeneinander abwägt. Dies würde außerdem eine genauere Beurteilung der Effizienz des Ansatzes erlauben.

2. Übertragung der Grundidee auf andere Arten von Anforderungsartefakten oder Systemen

Ein weiterer Punkt für zukünftige Forschungsarbeiten könnte die Übertragung der Grundidee des Ansatzes auf andere Arten von Anforderungsartefakten oder Systemen sein. Aufgrund der Wichtigkeit der Validierung von Anforderungen kann eine Sichtenbildung zur Unterstützung der Fehleridentifikation auch für andere Anforderungsmodelle wie bspw. Szenariomodelle von Interesse sein. In der Softwareproduktlinienentwicklung ist es üblich, Anforderungsartefakte zu Variabilitätsmodellen zu relationieren. Daher ergeben sich auch bei anderen Arten von Anforderungsartefakten ähnliche Probleme bzgl. der Validierung. Die Übertragung der Grundidee auf diese Arten von Anforderungsartefakten könnte die Validierung solcher Artefakte unterstützen.

Neben der Übertragung des Ansatzes auf andere Anforderungsartefakte ist auch seine Verwendung für die Entwicklung anderer Arten variabler Systeme denkbar. Produktlinien, Service-basierte oder selbst-adaptive Systeme ähneln bzgl. ihrer Rekonfigurierbarkeit Systems-of-Systems. Daher ist zu vermuten, dass bei der Validierung von Zielspezifikationen für solche Systeme ähnliche Herausforderungen existieren. Die Grundidee des Ansatzes könnte übertragbar sein.

Literaturverzeichnis

Abbott R (2006) *Open at the Top; Open at the Bottom; And Continually (but Slowly) Evolving.* In: 2006 IEEE/SMC International Conference on System of Systems Engineering. S. 1–6. doi:10.1109/SYSOSE.2006.1652271

Abeywickrama DB, Zambonelli F (2012) *Model Checking Goal-Oriented Requirements for Self-Adaptive Systems.* In: 2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems. S. 33–42. doi:10.1109/ECBS.2012.30

Ackoff RL (1971) *Towards a System of Systems Concepts.* In: Management Science 17(11): 661–671. doi:10.1287/mnsc.17.11.661

Ali R, Yu Y, Chitchyan R, Nhlabatsi A, Giorgini P (2009a) *Towards a Unified Framework for Contextual Variability in Requirements.* In: 2009 Third International Workshop on Software Product Management. S. 31–34. doi:10.1109/IWSPM.2009.8

Ali R, Dalpiaz F, Giorgini P (2009b) *A Goal Modeling Framework for Self-contextualizable Software.* In: Halpin T, Krogstie J, Nurcan S, Proper E, Schmidt R, Soffer P, Ukor R (Hrsg.) Enterprise, Business-Process and Information Systems Modeling. Springer Berlin Heidelberg, S. 326–338. doi:10.1007/978-3-642-01862-6_27

Ali R, Dalpiaz F, Giorgini P (2010) *A Goal-based Framework for Contextual Requirements Modeling and Analysis.* In: Requirements Engineering 15(4):439–458. doi:10.1007/s00766-010-0110-z

Ali R, Dalpiaz F, Giorgini P (2013) *Reasoning with Contextual Requirements: Detecting Inconsistency and Conflicts.* In: Information and Software Technology 55(1):35–57. doi:10.1016/j.infsof.2012.06.013

Almeida JRd, Camargo JB, Basseto BA, Paz SM (2003) *Best Practices in Code Inspection for Safety-critical Software.* In: IEEE Software 20(3):56–63. doi:10.1109/ms.2003.1196322

Amyot D, Ghanavati S, Horkoff J, Mussbacher G, Peyton L, Yu E (2010) *Evaluating Goal Models within the Goal-oriented Requirement Language.* In: International Journal of Intelligent Systems 25(8):841–877. doi:10.1002/int.20433

Anda AA, Amyot D (2019) *Arithmetic Semantics of Feature and Goal Models for Adaptive Cyber-Physical Systems.* In: 27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019. S. 245–256. doi:10.1109/RE.2019.00034

António S, Araújo J, Silva C (2009) *Adapting the I* Framework for Software Product Lines.* In: Heuser CA, Pernul G (Hrsg.) Advances in Conceptual Modeling - Challenging Perspectives. Springer Berlin Heidelberg, S. 286–295. doi:10.1007/978-3-642-04947-7_34

Apel S, Speidel H, Wendler P, Von Rhein A, Beyer D (2011) *Detection of Feature Interactions Using Feature-aware Verification.* In: 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011. S. 372–375. doi:10.1109/ASE.2011.6100075

Apel S, Kästner C (2009) *An Overview of Feature-Oriented Software Development.* In: The Journal of Object Technology 8(5):49–84. doi:10.5381/jot.2009.8.5.c5

Apel S, von Rhein A, Thüm T, Kästner C (2013) *Feature-interaction Detection Based on Feature-based Specifications.* In: Computer Networks 57(12):2399–2409. doi:10.1016/j.comnet.2013.02.025

Asadi M, Bagheri E, Gašević D, Hatala M, Mohabbati B (2011) *Goal-driven Software Product Line Engineering.* In: Proceedings of the 2011 ACM Symposium on Applied Computing. ACM, S. 691–698. doi:10.1145/1982185.1982336

Asadi M, Soltani S, Gasevic D, Hatala M, Bagheri E (2014) *Toward Automated Feature Model Configuration with Optimizing Non-functional Requirements.* In: Information and Software Technology 56(9):1144–1165. doi:10.1016/j.infsof.2014.03.005

Asadi M, Gröner G, Mohabbati B, Gašević D (2016) *Goal-oriented Modeling and Verification of Feature-oriented Product Lines.* In: Software & Systems Modeling 15(1): 257–279. doi:10.1007/s10270-014-0402-8

Aurum A, Petersson H, Wohlin C (2002) *State-of-the-art: Software Inspections After 25 Years.* In: Software Testing, Verification and Reliability 12(3):133–154. doi:10.1002/stvr.243

Bagheri E, Di Noia T, Ragone A, Gasevic D (2010) *Configuring Software Product Line Feature Models Based on Stakeholders' Soft and Hard Requirements.* In: Bosch J, Lee J (Hrsg.) Software Product Lines: Going Beyond. Springer Berlin Heidelberg, S. 16–31. doi:10.1007/978-3-642-15579-6_2

Bar-Yam Y, Allison MA, Batdorf R, Chen H, Generazio H, Singh H, Tucker S (2004) *The Characteristics and Emerging Behaviors of System of Systems.* In: NECSI: Complex Physical, Biological and Social Systems Project S. 1–16

Batory D (2005) *Feature Models, Grammars, and Propositional Formulas.* In: Obbink H, Pohl K (Hrsg.) Software Product Lines. Springer, S. 7–20. doi:10.1007/11554844_3

Bavota G, Gravino C, Oliveto R, De Lucia A, Tortora G, Genero M, Cruz-Lemus JA (2011) *Identifying the Weaknesses of UML Class Diagrams during Data Model*

Comprehension. In: Whittle J, Clark T, Kühne T (Hrsg.) Model Driven Engineering Languages and Systems. Springer, S. 168–182. doi:10.1007/978-3-642-24485-8_13

Benavides D, Trinidad P, Ruiz-Cortés A (2005) *Automated Reasoning on Feature Models.* In: Pastor O, Falcão e Cunha J (Hrsg.) Advanced Information Systems Engineering. Springer, S. 491–503. doi:10.1007/11431855_34

Benavides D, Segura S, Ruiz-Cortés A (2010) *Automated Analysis of Feature Models 20 Years Later: A Literature Review.* In: Information Systems 35(6):615–636. doi:10.1016/j.is.2010.01.001

Bencomo N, Hallsteinsen S, Almeida ESd (2012) *A View of the Dynamic Software Product Line Landscape.* In: Computer 45(10):36–41. doi:10.1109/MC.2012.292

Bennasri S, Souveyet C (2004) *Capturing Requirements Variability into Components - A Goal Driven Approach.* In: International Conference on Enterprise Information Systems. Scitepress, S. 438–443. doi:10.5220/0002638104380443

Berger T, Rublack R, Nair D, Atlee JM, Becker M, Czarnecki K, Wąsowski A (2013) *A Survey of Variability Modeling in Industrial Practice.* In: Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems. ACM, S. 1–8. doi:10.1145/2430502.2430513

Berling T, Runeson P (2003) *Evaluation of a Perspective Based Review Method Applied in an Industrial Setting.* In: IEE Proceedings - Software 150(3):177–184. doi:10.1049/ip-sen:20030483

Bidian C, Yu ESK (2007) *Towards Variability Design As Decision Boundary Placement.* In: Alves CF, Werneck V, Cysneiros LM (Hrsg.) Anais do WER07 - Workshop em Engenharia de Requisitos, Toronto, Canada, May 17-18, 2007. S. 139–148

Boardman J, Sauser B (2006) *System of Systems - the Meaning of Of.* In: 2006 IEEE/SMC International Conference on System of Systems Engineering. S. 118–123. doi:10.1109/SYSOSE.2006.1652284

Boehm B (1984) *Verifying and Validating Software Requirements and Design Specifications.* In: IEEE Software 1(1):75–88. doi:10.1109/MS.1984.233702

Borba C, Silva C (2009) *A Comparison of Goal-Oriented Approaches to Model Software Product Lines Variability.* In: Heuser CA, Pernul G (Hrsg.) Advances in Conceptual Modeling - Challenging Perspectives. Springer Berlin Heidelberg, S. 244–253. doi:10.1007/978-3-642-04947-7_30

Botangen KA, Yu J, Yongchareon S, Yang LH, Bai Q (2018) *Specifying and Reasoning About Contextual Preferences in the Goal-oriented Requirements Modelling.* In: Proceedings of the Australasian Computer Science Week Multiconference. ACM, S. 47:1–47:10. doi:10.1145/3167918.3167945

Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2004) *Tropos: An Agent-Oriented Software Development Methodology.* In: Autonomous Agents and Multi-Agent Systems 8(3):203–236. doi:10.1023/B:AGNT.0000018806.20944.ef

Brings J, Daun M (2019) *Towards Goal Modeling and Analysis for Networks of Collaborative Cyber-Physical Systems.* In: Proceedings of the ER Forum and Poster & Demos Session 2019 co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4, 2019. S. 70–83

Brings J, Daun M, Bandyszak T, Stricker V, Weyer T, Mirzaei E, Neumann M, Zernickel JS (2019) *Model-based Documentation of Dynamicity Constraints for Collaborative Cyber-Physical System Architectures: Findings from an Industrial Case Study.* In: Journal of Systems Architecture 97:153–167. doi:10.1016/j.sysarc.2019.02.012

Brings J, Daun M, Weyer T, Pohl K (2020a) *Goal-based Configuration Analysis for Networks of Collaborative Cyber-Physical Systems.* In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. ACM, S. 1387–1396. doi:10.1145/3341105.3374011

Brings J, Daun M, Weyer T, Pohl K (2020b) *Analyzing Goal Variability in Cyber-Physical System Networks.* In: ACM SIGAPP Applied Computing Review 20(2):19–35. doi:10.1145/3412816.3412818

Brunet J, Semmak F, Laleau R, Gnaho C (2008) *Using Variants in KAOS Goal Modelling.* In: ICEIS 2008 - Proceedings of the 10th International Conference on Enterprise Information Systems. S. 339–344

Bühne S, Lauenroth K, Pohl K (2005) *Modelling Requirements Variability across Product Lines.* In: 13th IEEE International Conference on Requirements Engineering (RE 2005), 29 August - 2 September 2005, Paris, France. S. 41–50. doi:10.1109/RE.2005.45

Campbell DT, Stanley JC (1963) *Experimental and Quasi-experimental Designs for Research.* Houghton Mifflin. ISBN:0-395-30787-2

Cavalcante E, Batista T, Bencomo N, Sawyer P (2015) *Revisiting Goal-Oriented Models for Self-Aware Systems-of-Systems.* In: 2015 IEEE International Conference on Autonomic Computing. S. 231–234. doi:10.1109/ICAC.2015.43

Chatzikonstantinou G, Kontogiannis K (2013) *Model Contextual Variability for Agents Using Goals and Commitments.* In: Proceedings of the 6th International i* Workshop. S. 103–108

Chen B, Peng X, Yu Y, Nuseibeh B, Zhao W (2014) *Self-adaptation Through Incremental Generative Model Transformations at Runtime.* In: Proceedings of the 36th International Conference on Software Engineering. ACM, S. 676–687. doi:10.1145/2568225.2568310

Chen P, Clothier J (2003) *Advancing Systems Engineering for Systems-of-Systems Challenges.* In: Systems Engineering 6(3):170–183. doi:10.1002/sys.10042

Chung L, Nixon BA, Yu E, Mylopoulos J (2000) *Non-Functional Requirements in Software Engineering.* Springer US. doi:10.1007/978-1-4615-5269-7

Clarke EM, Grumberg O, Peled DA (1999) *Model Checking.* MIT Press. ISBN:978-0-262-03270-4

Conradi R, Mohagheghi P, Arif T, Hegde LC, Bunde GA, Pedersen A (2003) *Object-Oriented Reading Techniques for Inspection of UML Models – An Industrial Experiment.* In: ECOOP 2003 – Object-Oriented Programming. Springer, Berlin, Heidelberg, S. 483–500. doi:10.1007/978-3-540-45070-2_21

Crnkovic I (2001) *Component-based Software Engineering — New Challenges in Software Development.* In: Software Focus 2(4):127–133. doi:10.1002/swf.45

Czarnecki K, Helsen S (2003) *Classification of Model Transformation Approaches.* In: OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture S. 1–17

Czarnecki K, Kim CHP (2005) *Cardinality-Based Feature Modeling and Constraints: A Progress Report.* In: OOPSLA'05 Workshop on Software Factories S. 1–9

Czarnecki K, Helsen S, Eisenecker U (2004) *Staged Configuration Using Feature Models.* In: Nord RL (Hrsg.) Software Product Lines. Springer, S. 266–283. doi:10.1007/978-3-540-28630-1_17

Czarnecki K, Helsen S, Eisenecker U (2005) *Formalizing Cardinality-based Feature Models and Their Specialization.* In: Software Process: Improvement and Practice 10(1):7–29. doi:10.1002/spip.213

Dardenne A, van Lamsweerde A, Fickas S (1993) *Goal-directed Requirements Acquisition.* In: Science of Computer Programming 20(1–2):3–50. doi:10.1016/0167-6423(93)90021-G

Daun M, Salmon A, Bandyszak T, Weyer T (2016) *Common Threats and Mitigation Strategies in Requirements Engineering Experiments with Student Participants.*

In: Requirements Engineering: Foundation for Software Quality 22nd International Working Conference, REFSQ 2016 , Gothenburg, Sweden, March 14-17, 2016. Proceedings. Springer, S. 269–285. doi:10.1007/978-3-319-30282-9_19

Daun M, Brings J, Aluko Obe P, Stenkova V (2021) *Reliability of Self-rated Experience and Confidence As Predictors for Students' Performance in Software Engineering.* In: Empirical Software Engineering 26(4):80. doi:10.1007/s10664-021-09972-6

de Mello RM, Teixeira EN, Schots M, Werner CML, Travassos GH (2014) *Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections.* In: Journal of Universal Computer Science 20(5):720–745. doi:10.3217/jucs-020-05-0720

Deb N, Chaki N, Ghose A (2016) *I*ToNuSMV: A Prototype for Enabling Model Checking of I* Models.* In: 24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016. S. 397–398. doi:10.1109/RE.2016.62

Deb N, Chaki N (2020) *Model Checking with I*.* In: Business Standard Compliance and Requirements Validation Using Goal Models S. 45–79. doi:10.1007/978-981-15-2501-8_4

DeVries B, Cheng BHC (2016) *Automatic Detection of Incomplete Requirements Via Symbolic Analysis.* In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. ACM, S. 385–395. doi:10.1145/2976767.2976791

DeVries B, Cheng BHC (2017a) *Using Models at Run Time to Detect Incomplete and Inconsistent Requirements.* In: MODELS (Satellite Events). S. 201–209

DeVries B, Cheng BHC (2017b) *Automatic Detection of Incomplete Requirements Using Symbolic Analysis and Evolutionary Computation.* In: Menzies T, Petke J (Hrsg.) Search Based Software Engineering. Springer International Publishing, S. 49–64. doi:10.1007/978-3-319-66299-2_4

Dhungana D, Falkner A, Haselbock A (2011) *Configuration of Cardinality-Based Feature Models Using Generative Constraint Satisfaction.* In: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications. S. 100–103. doi:10.1109/SEAA.2011.24

Duran MB, Mussbacher G, Thimmegowda N, Kienzle J (2015) *On the Reuse of Goal Models.* In: Fischer J, Scheidgen M, Schieferdecker I, Reed R (Hrsg.) SDL 2015: Model-Driven Engineering for Smart Cities. Springer International Publishing, S. 141–158. doi:10.1007/978-3-319-24912-4_11

Eisner H, Marciniak J, McMillan R (1991) *Computer-aided System of Systems (S2) Engineering.* In: Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, S. 531–537. doi:10.1109/ICSMC.1991.169739

Elahi G, Yu E (2011) *Requirements Trade-offs Analysis in the Absence of Quantitative Measures: A Heuristic Method.* In: Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11). S. 651–658. doi:10.1145/1982185.1982331

Fagan M (1976) *Design and Code Inspections to Reduce Errors in Program Development.* In: IBM Systems Journal 15(3):182–211. doi:10.1147/sj.153.0182

Field A (2013) *Discovering Statistics Using SPSS.* 4th Edition, Sage Publications Ltd. ISBN:978-93-5150-082-7

Finkelstein A, Goedicke M, Kramer J, Niskier C (1991) *ViewPoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering.* In: Algebraic Methods II: Theory, Tools and Applications. Springer, Berlin, Heidelberg, S. 29–54. doi:10.1007/3-540-53912-3_17

Fuxman A, Pistore M, Mylopoulos J, Traverso P (2000) *Model Checking Early Requirements Specifications in Tropos.* In: Proceedings Fifth IEEE International Symposium on Requirements Engineering. IEEE, S. 174–181. doi:10.1109/ISRE.2001.948557

Fuxman A, Liu L, Mylopoulos J, Pistore M, Roveri M, Traverso P (2004) *Specifying and Analyzing Early Requirements in Tropos.* In: Requirements Engineering 9(2):132–150. doi:10.1007/s00766-004-0191-7

Galindo JA, Benavides D, Trinidad P, Gutiérrez-Fernández AM, Ruiz-Cortés A (2019) *Automated Analysis of Feature Models: Quo Vadis?* In: Computing 101(5):387–433. doi:10.1007/s00607-018-0646-1

Garcés L, Nakagawa E (2017) *A Process to Establish, Model and Validate Missions of Systems-of-systems in Reference Architectures.* In: Proceedings of the Symposium on Applied Computing. ACM, S. 1765–1772. doi:10.1145/3019612.3019799

Garro A, Tundis A (2015) *On the Reliability Analysis of Systems and SoS: The RAMSAS Method and Related Extensions.* In: IEEE Systems Journal 9(1):232–241. doi:10.1109/JSYST.2014.2321617

Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2002) *Reasoning with Goal Models.* In: Goos G, Hartmanis J, van Leeuwen J, Spaccapietra S, March ST, Kambayashi Y (Hrsg.) Conceptual Modeling — ER 2002, S. 167–181. Springer Berlin Heidelberg. doi:10.1007/3-540-45816-6_22

Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2003) *Formal Reasoning Techniques for Goal Models.* In: Goos G, Hartmanis J, van Leeuwen J, Spaccapietra S, March S, Aberer K (Hrsg.) Journal on Data Semantics I, S. 1–20. Springer Berlin Heidelberg. doi:10.1007/978-3-540-39733-5_1

Giorgini P, Massacci F, Mylopoulos J, Zannone N (2004) *Requirements Engineering Meets Trust Management - Model, Methodology, and Reasoning.* In: In Proceedings of the 2nd International Conference on Trust Management. Springer Berlin Heidelberg, S. 176–190. doi:10.1007/978-3-540-24747-0_14

Giorgini P, Mylopoulos J, Sebastiani R (2005) *Goal-oriented Requirements Analysis and Reasoning in the Tropos Methodology.* In: Engineering Applications of Artificial Intelligence 18(2):159–171. doi:10.1016/j.engappai.2004.11.017

Goldsby HJ, Sawyer P, Bencomo N, Cheng BHC, Hughes D (2008) *Goal-Based Modeling of Dynamically Adaptive System Requirements.* In: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008). S. 36–45. doi:10.1109/ECBS.2008.22

Gonzales-Baixauli B, Leite JCSP, Mylopoulos J (2004) *Visual Variability Analysis for Goal Models.* In: 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan. S. 198–207. doi:10.1109/ICRE.2004.1335677

González-Baixauli B, Laguna MA (2007) *Using Goal-Models to Analyze Variability.* In: First International Workshop on Variability Modelling of Software-Intensive Systems, VaMoS 2007, Limerick, Ireland, January 16-18, 2007. Proceedings. S. 101–107

Grubb AM, Chechik M (may 2021) *Formal Reasoning for Analyzing Goal Models That Evolve Over Time.* In: Requirements Engineering 26(3):423–457. doi:10.1007/s00766-021-00350-8

Guedes G, Silva C, Castro J (2011) *Goals and Scenarios for Requirements Engineering of Software Product Lines.* In: Proceedings of the 5th International i* Workshop. S. 108–113

Guedes G, Silva C, Castro J (2013) *Goals and Scenarios to Software Product Lines: The GS2SPL Approach.* In: Proceedings of Requirements Engineering@Brazil 2013

Guedes G, Silva C, Soares M (2017) *Comparing Configuration Approaches for Dynamic Software Product Lines.* In: Proceedings of the 31st Brazilian Symposium on Software Engineering. ACM, S. 134–143. doi:10.1145/3131151.3131162

Gámez N, Fuentes L, Troya JM (2015) *Creating Self-Adapting Mobile Systems with Dynamic Software Product Lines.* In: IEEE Software 32(2):105–112. doi:10.1109/MS.2014.24

Hallsteinsen S, Stav E, Solberg A, Floch J (2006) *Using Product Line Techniques to Build Adaptive Systems.* In: 10th International Software Product Line Conference (SPLC'06). S. 141–150. doi:10.1109/SPLINE.2006.1691586

Hallsteinsen S, Hinchey M, Park S, Schmid K (2008) *Dynamic Software Product Lines.* In: Computer 41(4):93–95. doi:10.1109/MC.2008.123

Hassine J, Amyot D (2016) *A Questionnaire-based Survey Methodology for Systematically Validating Goal-oriented Models.* In: Requirements Engineering 21(2):285–308. doi:10.1007/s00766-015-0221-7

Hassine J, Amyot D (2017) *An Empirical Approach toward the Resolution of Conflicts in Goal-oriented Models.* In: Software & Systems Modeling 16(1):279–306. doi:10.1007/s10270-015-0460-6

Horkoff J, Yu E (2009) *Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences.* In: Persson A, Stirna J (Hrsg.) The Practice of Enterprise Modeling. Springer, S. 145–160. doi:10.1007/978-3-642-05352-8_12

Horkoff J, Yu E (2010) *Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach.* In: Parsons J, Saeki M, Shoval P, Woo C, Wand Y (Hrsg.) Conceptual Modeling – ER 2010. Springer, S. 59–75. doi:10.1007/978-3-642-16373-9_5

Horkoff J, Yu E (2011) *Analyzing Goal Models: Different Approaches and How to Choose among Them.* In: Proceedings of the 2011 ACM Symposium on Applied Computing - SAC '11. ACM Press, S. 675–682. doi:10.1145/1982185.1982334

Horkoff J, Yu E (2013) *Comparison and Evaluation of Goal-oriented Satisfaction Analysis Techniques.* In: Requirements Engineering 18(3):199–222. doi:10.1007/s00766-011-0143-y

Horkoff J, Yu E (2016) *Interactive Goal Model Analysis for Early Requirements Engineering.* In: Requirements Engineering 21(1):29–61. doi:10.1007/s00766-014-0209-8

Horkoff J, Aydemir FB, Cardoso E, Li T, Maté A, Paja E, Salnitri M, Piras L, Mylopoulos J, Giorgini P (2019) *Goal-oriented Requirements Engineering: An Extended Systematic Mapping Study.* In: Requirements Engineering 24(2):133–160. doi:10.1007/s00766-017-0280-z

Hui B, Liaskos S, Mylopoulos J (2003) *Requirements Analysis for Customizable Software: A Goals-skills-preferences Framework.* In: 11th IEEE International Conference on Requirements Engineering (RE 2003), 8-12 September 2003, Monterey Bay, CA, USA. S. 117–126. doi:10.1109/ICRE.2003.1232743

International Telecommunication Union (2018) *Recommendation Z.151 (10/18): User Requirements Notation (URN) - Language Definition*, 2018

ISO/IEC/IEEE 15288 (2015) *ISO/IEC/IEEE International Standard - Systems and Software Engineering – System Life Cycle Processes*, 2015

ISO/IEC/IEEE 24765 (2017) *ISO/IEC/IEEE International Standard - Systems and Software Engineering–Vocabulary*, 2017

ISO/IEC/IEEE 42010 (2011) *ISO/IEC/IEEE Systems and Software Engineering – Architecture Description*, 2011

Jamshidi M (2008) *System of Systems Engineering - New Challenges for the 21st Century.* In: IEEE Aerospace and Electronic Systems Magazine 23(5):4–19. doi:10.1109/MAES.2008.4523909

Jarzabek S, Yang B, Yoeun S (2006) *Addressing Quality Attributes in Domain Analysis for Product Lines.* In: IEE Proceedings - Software 153(2):61–73. doi:10.1049/ip-sen:20050008

Jureta IJ, Borgida A, Ernst NA, Mylopoulos J (2010) *Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling.* In: RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010. S. 115–124. doi:10.1109/RE.2010.24

Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS (1990) *Feature-oriented Domain Analysis (FODA) Feasibility Study.* Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, 1990

Keating C, Rogers R, Unal R, Dryer D, Sousa-Poza A, Safford R, Peterson W, Rabadi G (2003) *System of Systems Engineering.* In: Engineering Management Journal 15(3):36–45. doi:10.1080/10429247.2003.11415214

Kim J, Kim M, Yang H, Park S (2004) *A Method and Tool Support for Variant Requirements Analysis: Goal and Scenario Based Approach.* In: 11th Asia-Pacific Software Engineering Conference. S. 168–175. doi:10.1109/APSEC.2004.4

Kim K, Kim H, Kim S, Chang G (2008) *A Case Study on SW Product Line Architecture Evaluation: Experience in the Consumer Electronics Domain.* In: 2008 The Third International Conference on Software Engineering Advances. S. 192–197. doi:10.1109/ICSEA.2008.80

Kollanus S, Koskinen J (2009) *Survey of Software Inspection Research.* In: The Open Software Engineering Journal 3(1). doi:10.2174/1874107x00903010015

Kopetz H, Bondavalli A, Brancati F, Frömel B, Höftberger O, Iacob S (2016) *Emergence in Cyber-Physical Systems-of-Systems (CPSoSs).* In: Bondavalli A, Bouchenak S, Kopetz H (Hrsg.) Cyber-Physical Systems of Systems: Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy, S. 73–96. Springer International Publishing. doi:10.1007/978-3-319-47590-5_3

Kotonya G, Sommerville I (1996) *Requirements Engineering with Viewpoints.* In: Software Engineering Journal 11(1):5–18. doi:10.1049/sej.1996.0002

Kotov V (1999) *Systems of Systems As Communicating Structures.* In: Object-Oriented Technology and Computing Systems Re-engineering, S. 141–154. Elsevier. doi:10.1533/9781782420613.141

Krygiel AJ (1999) *Behind the Wizard's Curtain: An Integration Environment for a System of Systems.* National Defense University Press. ISBN:978-1-57906-018-3

Laitenberger O (1998) *Studying the Effects of Code Inspection and Structural Testing on Software Quality.* In: Proceedings Ninth International Symposium on Software Reliability Engineering. S. 237–246. doi:10.1109/issre.1998.730887

Laitenberger O, Atkinson C, Schlich M, El Emam K (2000) *An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents.* In: Journal of Systems and Software 53(2):183–204. doi:10.1016/S0164-1212(00)00052-2

Lamsweerde Av (2001) *Goal-oriented Requirements Engineering: A Guided Tour.* In: 5th IEEE International Symposium on Requirements Engineering (RE 2001), 27-31 August 2001, Toronto, Canada. S. 249–262. doi:10.1109/ISRE.2001.948567

Lamsweerde Av (2009) *Requirements Engineering: From System Goals to UML Models to Software Specifications.* John Wiley & Sons. ISBN:978-0-470-01270-3

Lapouchnian A, Mylopoulos J (2011) *Capturing Contextual Variability in I* Models.* In: Proceedings of the 5th International i* Workshop. S. 96–101

Lapouchnian A, Mylopoulos J (2009) *Modeling Domain Variability in Requirements Engineering with Contexts.* In: Laender AHF, Castano S, Dayal U, Casati F, de Oliveira JPM (Hrsg.) Conceptual Modeling - ER 2009. Springer Berlin Heidelberg, S. 115–130. doi:10.1007/978-3-642-04840-1_11

Lapouchnian A, Yu Y, Liaskos S, Mylopoulos J (2006) *Requirements-Driven Design of Autonomic Application Software.* In: Proceedings of the Annual International Conference on Computer Science and Software Engineering. ACM, S. 23–37. doi:10.1145/1188966.1188976

Lapouchnian A, Yu Y, Mylopoulos J (2007) *Requirements-Driven Design and Configuration Management of Business Processes.* In: Alonso G, Dadam P, Rosemann M (Hrsg.) Business Process Management. Springer Berlin Heidelberg, S. 246–261. doi:10.1007/978-3-540-75183-0_18

Lee J, Kang KC, Sawyer P, Lee H (2014) *A Holistic Approach to Feature Modeling for Product Line Requirements Engineering.* In: Requirements Engineering 19(4):377–395. doi:10.1007/s00766-013-0183-6

Letier E, van Lamsweerde A (2004) *Reasoning About Partial Goal Satisfaction for Requirements and Design Engineering.* In: Proceedings of the 12th International Symposium on Foundations of Software Engineering. ACM, S. 53–62. doi:10.1145/1029894.1029905

Lewis GA, Morris E, Place P, Simanta S, Smith DB (2009) *Requirements Engineering for Systems of Systems.* In: 2009 3rd Annual IEEE Systems Conference. S. 247–252. doi:10.1109/SYSTEMS.2009.4815806

Lewis G, Morris E, Simanta S, Smith D (2011) *Service Orientation and Systems of Systems.* In: IEEE Software 28(1):58–63. doi:10.1109/MS.2011.15

Liaskos S, Rogozhkin V (2011) *Applying Preference-based Customization.* In: RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011. S. 353–354. doi:10.1109/RE.2011.6051672

Liaskos S, Lapouchnian A, Yiqiao Wang , Yijun Yu , Easterbrook S (2005) *Configuring Common Personal Software: A Requirements-driven Approach.* In: 13th IEEE International Conference on Requirements Engineering (RE 2005), 29 August - 2 September 2005, Paris, France. S. 9–18. doi:10.1109/RE.2005.19

Liaskos S, McIlraith SA, Sohrabi S, Mylopoulos J (2010) *Integrating Preferences into Goal Models for Requirements Engineering.* In: RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010. S. 135–144. doi:10.1109/RE.2010.26

Liaskos S, Jiang L, Lapouchnian A, Wang Y, Yu Y, Mylopoulos J (2007) *Exploring the Dimensions of Variability: A Requirements Engineering Perspective.* In: First International Workshop on Variability Modelling of Software-Intensive Systems, VaMoS 2007, Limerick, Ireland, January 16-18, 2007. Proceedings. S. 10

Liaskos S, McIlraith SA, Mylopoulos J (2009) *Towards Augmenting Requirements Models with Preferences.* In: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE, S. 565–569. doi:10.1109/ASE.2009.91

Liaskos S, Litoiu M, Jungblut MD, Mylopoulos J (2011a) *Goal-Based Behavioral Customization of Information Systems.* In: Mouratidis H, Rolland C (Hrsg.) Advanced Information Systems Engineering. Springer Berlin Heidelberg, S. 77–92. doi:10.1007/978-3-642-21640-4_8

Liaskos S, McIlraith SA, Sohrabi S, Mylopoulos J (2011b) *Representing and Reasoning about Preferences in Requirements Engineering.* In: Requirements Engineering 16(3):227. doi:10.1007/s00766-011-0129-9

Liaskos S, Khan SM, Litoiu M, Jungblut MD, Rogozhkin V, Mylopoulos J (2012) *Behavioral Adaptation of Information Systems through Goal Models.* In: Information Systems 37(8):767–783. doi:10.1016/j.is.2012.05.006

Liu Y, Su Y, Yin X, Mussbacher G (2014a) *Combined Propagation-based Reasoning with Goal and Feature Models.* In: IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014. S. 27–36. doi:10.1109/MoDRE.2014.6890823

Liu Y, Su Y, Yin X, Mussbacher G (2014b) *Combined Goal and Feature Model Reasoning with the User Requirements Notation and jUCMNav.* In: 2014 IEEE 22nd International Requirements Engineering Conference (RE). S. 321–322. doi:10.1109/RE.2014.6912277

Lucia AD, Gravino C, Oliveto R, Tortora G (2008) *Data Model Comprehension: An Empirical Comparison of ER and UML Class Diagrams.* In: 2008 16th IEEE International Conference on Program Comprehension. S. 93–102. doi:10.1109/icpc.2008.26

Lukasik SJ (1998) *Systems, Systems of Systems, and the Education of Engineers.* In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(1):55–60. doi:10.1017/S0890060498121078

Maiden N, Lockerbie J, Randall D, Jones S, Bush D (2007) *Using Satisfaction Arguments to Enhance I* Modelling of an Air Traffic Management System.* In: 23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015. S. 49–52. doi:10.1109/RE.2007.14

Maier MW (1998) *Architecting Principles for Systems-of-systems.* In: Systems Engineering 1 (4):267–284. doi:10.1002/(sici)1520-6858(1998)1:4<267::aid-sys3>3.0.co;2-d

Maldonado JC, Carver J, Shull F, Sandra Camargo Pinto Ferraz Fabbri , Dória E, Martimiano LAF, Mendonça MG, Basili VR (2006) *Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness.* In: Empirical Software Engineering 11(1):119–142. doi:10.1007/s10664-006-5967-6

Maßen T, Lichter H (2004) *Deficiencies in Feature Models.* In: Variability Management for Product Derivation – Towards Tool Support: SPLC 2004 Workshop

Metzger A, Pohl K, Heymans P, Schobbens P, Saval G (2007) *Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis.* In: 15th IEEE International Requirements Engineering Conference, RE 2007, October 15-19th, 2007, New Delhi, India. S. 243–253. doi:10.1109/RE.2007.61

Miller J, Wood M, Roper M (1998) *Further Experiences with Scenarios and Checklists.* In: Empirical Software Engineering 3(1):37–64. doi:10.1023/A:1009735805377

Morandini M, Penserini L, Perini A, Marchetto A (2017) *Engineering Requirements for Adaptive Systems.* In: Requirements Engineering 22(1):77–103. doi:10.1007/s00766-015-0236-0

Mussbacher G (2008) *Aspect-Oriented User Requirements Notation: Aspects in Goal and Scenario Models.* In: Giese H (Hrsg.) Models in Software Engineering. Springer Berlin Heidelberg, S. 305–316. doi:10.1007/978-3-540-69073-3_32

Mussbacher G, Araújo J, Moreira A, Amyot D (2012) *AoURN-based Modeling and Analysis of Software Product Lines.* In: Software Quality Journal 20(3):645–687. doi:10.1007/s11219-011-9153-8

Mylopoulos J, Chung L, Nixon B (1992) *Representing and Using Nonfunctional Requirements: A Process-oriented Approach.* In: IEEE Transactions on Software Engineering 18(6):483–497. doi:10.1109/32.142871

Nakagawa H, Ohsuga A, Honiden S (2011) *GOCC: A Configuration Compiler for Self-adaptive Systems Using Goal-oriented Requirements Description.* In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, S. 40–49. doi:10.1145/1988008.1988015

Nielsen CB, Larsen PG, Fitzgerald J, Woodcock J, Peleska J (2015) *Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions.* In: ACM Computing Surveys 48(2):18:1–18:41. doi:10.1145/2794381

Noam EM (1994) *Beyond Liberalization: From the Network of Networks to the System of Systems.* In: Telecommunications Policy 18(4):286–294. doi:10.1016/0308-5961(94)90002-7

Noorian M, Bagheri E, Du W (2014) *From Intentions to Decisions: Understanding Stakeholders' Objectives in Software Product Line Configuration.* In: International Conference on Software Engineering & Knowledge Engineering S. 671–677

Noorian M, Bagheri E, Du W (2017) *Toward Automated Quality-centric Product Line Configuration Using Intentional Variability.* In: Journal of Software: Evolution and Process 29(9):e1870. doi:10.1002/smr.1870

Nuseibeh B, Kramer J, Finkelstein A (1994) *A Framework for Expressing the Relationships between Multiple Views in Requirements Specification.* In: IEEE Transactions on Software Engineering 20(10):760–773. doi:10.1109/32.328995

Ogawa H, Kumeno F, Honiden S (2008) *Model Checking Process with Goal Oriented Requirements Analysis.* In: 2008 15th Asia-Pacific Software Engineering Conference. S. 377–384. doi:10.1109/APSEC.2008.71

Park S, Kim M, Sugumaran V (2004) *A Scenario, Goal and Feature-oriented Domain Analysis Approach for Developing Software Product Lines.* In: Industrial Management & Data Systems 104(4):296–308. doi:10.1108/02635570410530711

Pei RS (2000) *System of Systems Integration (SoSI)-A "SMART"Way of Acquiring Army C412WS Systems.* In: Summer Computer Simulation Conference. Society for Computer Simulation International; 1998, S. 574–579

Peng X, Chen B, Yu Y, Zhao W (2012) *Self-tuning of Software Systems through Dynamic Quality Tradeoff and Value-based Feedback Control Loop.* In: Journal of Systems and Software 85(12):2707–2719. doi:10.1016/j.jss.2012.04.079

Penserini L, Perini A, Susi A, Mylopoulos J (2007) *High Variability Design for Software Agents: Extending Tropos.* In: ACM Transactions on Autonomous and Adaptive Systems 2 (4):16. doi:10.1145/1293731.1293736

Pohl K, Böckle G, Linden Fvd (2005) *Software Product Line Engineering - Foundations, Principles, and Techniques.* Springer. doi:10.1007/3-540-28901-1

Rohleder C (2008) *Visualizing the Impact of 4on-Functional Requirements on Variants : A Case Study.* In: 2008 Requirements Engineering Visualization. S. 11–20. doi:10.1109/REV.2008.7

Runeson P, Höst M, Rainer A, Regnell B (2012) *Case Study Research in Software Engineering: Guidelines and Examples.* John Wiley & Sons, Inc. doi:10.1002/9781118181034

Sage AP, Cuppan CD (2001) *On the Systems Engineering and Management of Systems of Systems and Federations of Systems.* In: Information-Knowledge-Systems Management 2(4): 325–345. doi:10.5555/1234195.1234200

Salehie M, Pasquale L, Omoronyia I, Ali R, Nuseibeh B (2012) *Requirements-driven Adaptive Security: Protecting Variable Assets at Runtime.* In: 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA, September 24-28, 2012. S. 111–120. doi:10.1109/RE.2012.6345794

Salman I, Misirli AT, Juristo N (2015) *Are Students Representatives of Professionals in Software Engineering Experiments?* In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). S. 666–676. doi:10.1109/ICSE.2015.82

Santos E, Castro J, Sanchez J, Pastor O (2010) *A Goal-Oriented Approach for Variability in BPMN.* In: Anais do WER10 - Workshop em Engenharia de Requisitos, Cuenca, Ecuador, April 12-13, 2010

Schnabel T, Weckesser M, Kluge R, Lochau M, Schürr A (2016) *CardyGAn: Tool Support for Cardinality-based Feature Models.* In: Proceedings of the Tenth International Workshop

on Variability Modelling of Software-intensive Systems. ACM, S. 33–40. doi:10.1145/2866614.2866619

Schobbens PY, Heymans P, Trigaux JC (2006) *Feature Diagrams: A Survey and a Formal Semantics.* In: 14th IEEE International Conference on Requirements Engineering (RE 2006), 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA. S. 139–148. doi:10.1109/RE.2006.23

Sebastiani R, Giorgini P, Mylopoulos J (2004) *Simple and Minimum-Cost Satisfiability for Goal Models.* In: Persson A, Stirna J (Hrsg.) Advanced Information Systems Engineering. Springer Berlin Heidelberg, S. 20–35. doi:10.1007/978-3-540-25975-6_4

Segura S (2008) *Automated Analysis of Feature Models Using Atomic Sets.* In: Software Product Lines, 12th International Conference, SPLC 2008, Limerick, Ireland, September 8-12, 2008, Proceedings. Second Volume (Workshops). S. 201–207

Semmak F, Laleau R, Gnaho C (2009) *Supporting Variability in Goal-based Requirements.* In: 2009 Third International Conference on Research Challenges in Information Science. S. 237–246. doi:10.1109/RCIS.2009.5089287

Semmak F, Brunet J (2006) *Variability in Goal-Oriented Domain Requirements.* In: Morisio M (Hrsg.) Reuse of Off-the-Shelf Components. Springer Berlin Heidelberg, S. 390–394. doi:10.1007/11763864_30

Semmak F, Gnaho C, Laleau R (2008) *Extended KAOS to Support Variability for Goal Oriented Requirements Reuse.* In: Model Driven Information Systems Engineering: Enterprise, User and System Models. S. 22–33

Sendall S, Kozaczynski W (2003) *Model Transformation: The Heart and Soul of Model-driven Software Development.* In: IEEE Software 20(5):42–45. doi:10.1109/MS.2003.1231150

Shamsaei A, Amyot D, Pourshahid A, Braun E, Yu E, Mussbacher G, Tawhid R, Cartwright N (2013) *An Approach to Specify and Analyze Goal Model Families.* In: Haugen O, Reed R, Gotzhein R (Hrsg.) System Analysis and Modeling: Theory and Practice. Springer, S. 34–52. doi:10.1007/978-3-642-36757-1_3

Sharawi A, Sala-Diakanda SN, Dalton A, Quijada S, Yousef N, Rabelo L, Sepúlveda J (2006) *A Distributed Simulation Approach for Modeling and Analyzing Systems of Systems.* In: Proceedings of the 38th Conference on Winter Simulation. S. 1028–1035. doi:10.1109/WSC.2006.323191

Shenhar A (jul 1994) *A New Systems Engineering Taxonomy.* In: Proceedings of the 4th International Symposium of the National Council on System Engineering. Wiley, S. 261–276

Siegmund J, Siegmund N, Apel S (2015) *Views on Internal and External Validity in Empirical Software Engineering.* In: Bertolino A, Canfora G, Elbaum SG (Hrsg.) 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1. IEEE, S. 9–19. doi:10.1109/ICSE.2015.24

Silva C, Alencar F, Araújo J, Moreira A, Castro J (2008) *Tailoring an Aspectual Goal-Oriented Approach to Model Features.* In: Proceedings of the Twentieth International Conference on Software Engineering Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1-3, 2008 S. 472–477

Silva C, Borba C, Castro J (2011) *A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines.* In: Anais do WER11 - Workshop em Engenharia de Requisitos, Rio de Janeiro-RJ, Brasil, Abril 28-29, 2011

Silva E, Batista T, Oquendo F (2015) *A Mission-oriented Approach for Designing System-of-systems.* In: 2015 10th System of Systems Engineering Conference (SoSE). IEEE, S. 346–351. doi:10.1109/SYSOSE.2015.7151951

Silva E, Batista T (2018) *Formal Modeling Systems-of-systems Missions with mKAOS.* In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. ACM, S. 1674–1679. doi:10.1145/3167132.3167311

Silva E, Cavalcante E, Batista T, Oquendo F, Delicato FC, Pires PF (2014) *On the Characterization of Missions of Systems-of-Systems.* In: Proceedings of the 2014 European Conference on Software Architecture Workshops. ACM, S. 1–8. doi:10.1145/2642803.2642829

Soltani S, Asadi M, Gašević D, Hatala M, Bagheri E (2012) *Automated Planning for Feature Model Configuration Based on Functional and Non-functional Requirements.* In: Proceedings of the 16th International Software Product Line Conference - Volume 1. ACM, S. 56–65. doi:10.1145/2362536.2362548

Spillner A, Linz T (2019) *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester – Foundation Level nach ISTQB®-Standard.* 6., überarbeitete und aktualisierte Edition, dpunkt.verlag GmbH. ISBN:9783864905834

Stein J, Nunes I, Cirilo E (2014) *Preference-based Feature Model Configuration with Multiple Stakeholders.* In: Proceedings of the 18th International Software Product Line Conference - Volume 1. ACM, S. 132–141. doi:10.1145/2648511.2648525

Than Tun T, Boucher Q, Classen A, Hubaux A, Heymans P (2009) *Relating Requirements and Feature Configurations: A Systematic Approach.* In: Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, S. 201–210

Trinidad P, Benavides D, Durán A, Ruiz-Cortés A, Toro M (2008) *Automated Error Analysis for the Agilization of Feature Modeling.* In: Journal of Systems and Software 81(6): 883–896. doi:10.1016/j.jss.2007.10.030

Trinidad P, Benavides D, Ruiz-Cortes A (2004) *Improving Decision Making in Software Product Lines Product Plan Management.* In: Proceedings of the V ADIS 2004 Workshop on Decision Support in Software Engineering

Turner M, Budgen D, Brereton P (2003) *Turning Software into a Service.* In: Computer 36 (10):38–44. doi:10.1109/MC.2003.1236470

Uchitel S, Chatley R, Kramer J, Magee J (2006) *Goal and Scenario Validation: A Fluent Combination.* In: Requirements Engineering 11(2):123–137. doi:10.1007/s00766-005-0024-3

Valipour MH, Amirzafari B, Maleki KN, Daneshpour N (2009) *A Brief Survey of Software Architecture Concepts and Service Oriented Architecture.* In: 2009 2nd IEEE International Conference on Computer Science and Information Technology. S. 34–38. doi:10.1109/ICCSIT.2009.5235004

van Deursen A, Klint P (2004) *Domain-Specific Language Design Requires Feature Descriptions.* In: Journal of Computing and Information Technology 10(1):20. doi:10.2498/cit.2002.01.01

van Lamsweerde A (2009) *Reasoning About Alternative Requirements Options.* In: Borgida AT, Chaudhri VK, Giorgini P, Yu ES (Hrsg.) Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos, S. 380–397. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02463-4_20

Wang H, Mehta R, Supakkul S, Chung L (2011) *Rule-based Context-aware Adaptation Using a Goal-oriented Ontology.* In: Proceedings of the 2011 International Workshop on Situation Activity & Goal Awareness. ACM, S. 67–76. doi:10.1145/2030045.2030061

Wang H, Mehta R, Chung L, Supakkul S, Huang L (2012) *Rule-based Context-aware Adaptation: A Goal-oriented Approach.* In: International Journal of Pervasive Computing and Communications doi:10.1108/17427371211262662

Weckesser M, Lochau M, Schnabel T, Richerzhagen B, Schürr A (2016) *Mind the Gap! Automated Anomaly Detection for Potentially Unbounded Cardinality-Based Feature Models.* In: Fundamental Approaches to Software Engineering. Springer, Berlin, Heidelberg, S. 158–175. doi:10.1007/978-3-662-49665-7_10

Wohlin C (2012) *Experimentation in Software Engineering.* Springer. ISBN:978-3-642-29044-2 978-3-642-29043-5

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) *Experimentation in Software Engineering.* 2012 edition, Springer. ISBN:978-3-642-29043-5

Yu ESK (1996) *Modelling Strategic Relationships for Process Reengineering.* PhD thesis, University of Toronto

Yu E (1997) *Towards Modelling and Reasoning Support for Early-phase Requirements Engineering.* In: , Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997. S. 226–235. doi:10.1109/ISRE.1997.566873

Zave P, Jackson M (1997) *Four Dark Corners of Requirements Engineering.* In: ACM Transactions on Software Engineering and Methodology 6(1):1–30. doi:10.1145/237432.237434

Zhang W, Zhao H, Mei H (2004) *A Propositional Logic-Based Method for Verification of Feature Models.* In: Davies J, Schulte W, Barnett M (Hrsg.) Formal Methods and Software Engineering. Springer, S. 115–130. doi:10.1007/978-3-540-30482-1_16

Anhang

soSci
oFb - der onlineFragebogen

Seite 01

Herzlich Willkommen! Vielen Dank für Ihre Teilnahme!

Vielen Dank für Ihre Teilnahme an diesem Experiment. Das Experiment beschäftigt sich mit dem manuellen Review modellbasierter Spezifikationen gegen Stakeholder Wünsche.

Das Experiment wird etwa 30 min dauern. Bitte planen Sie für diese Zeit keine weiteren Tätigkeiten ein, da wir auch die Bearbeitungszeit messen werden.

Bitte benutzen Sie nicht den Zurück-Button ihres Browsers, da es für uns wichtig ist, dass Sie die einzelnen Fragen der Reihe nach, ohne Rückschritte, beantworten.

Seite 02

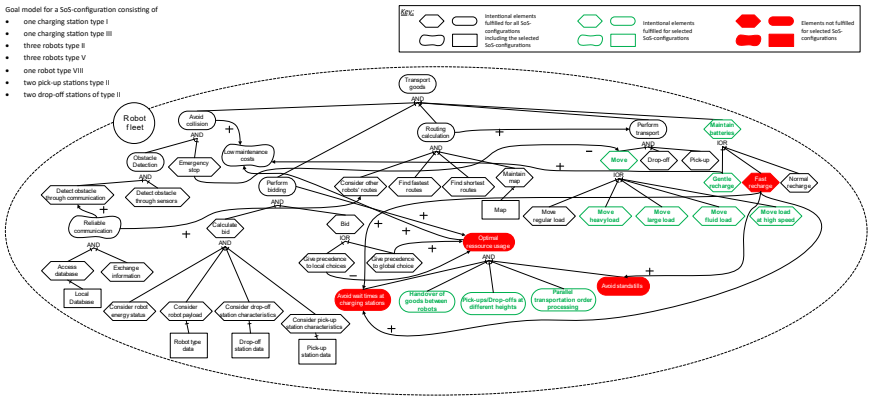
Review von modellbasierten Spezifikationen

Auf den nächsten beiden Seiten werden Sie Modelle finden, die wir Sie bitten möchten gegen textuelle Aussagen zu überprüfen. Im Detail finden Sie
a) ein SoS-Zielmodell, dass die Ziele eines Systems, die möglichen Zusammensetzungen dieses Systems und die Zusammenhänge zwischen Zielen und möglichen Zusammensetzungen zeigt
b) Sichten, die entweder die Ziele einer bestimmten Zusammensetzung zeigen oder die Zusammensetzungen zeigen, die bestimmte Ziele erreichen können

Unter den Spezifikationen finden Sie Aussagen. Wir möchten Sie bitten, die dargestellten Modelle zu reviewen und zu entscheiden, ob die textuellen Aussagen in den Spezifikationen dargestellt sind. Bitte geben Sie auch Ihr Selbstvertrauen (Confidence) bei jeder einzelnen Entscheidung an.

Zielsicht

Nachfolgend ist eine Sicht auf das Zielmodell einer Roboterflotte gegeben. Diese Sicht stellt dar welche Ziele, die Roboterflotte in einer bestimmten Zusammensetzung erreichen kann. Bitte führen Sie jetzt einen Review dieser Sicht durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

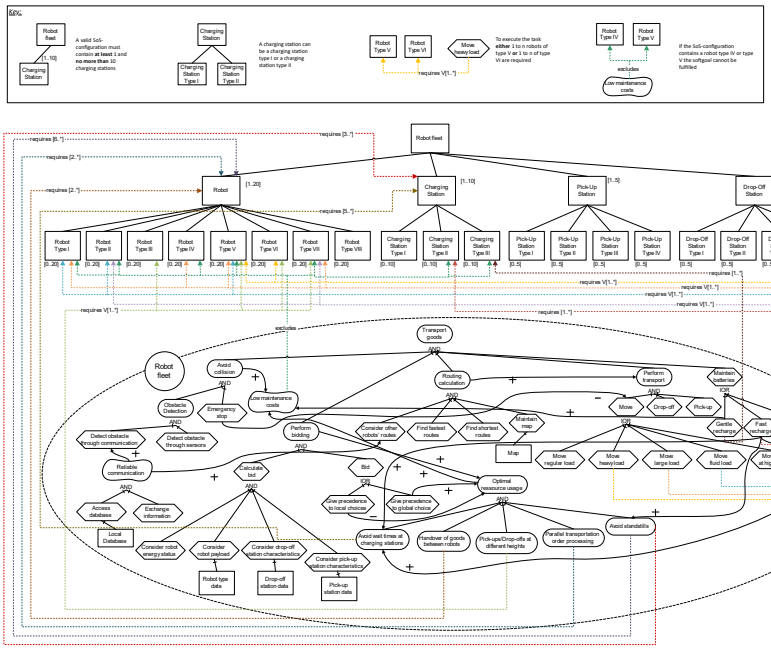
An SoS-configuration consisting of:

- one charging station type I
- one charging station type III
- three robots type II
- three robots type V
- one robot type VIII
- two pick-up stations type II
- two drop-off stations of type II

	Dargestellt	Nicht dargestellt	sehr sicher ◯ ◯ ◯ ◯ ◯ sehr unsicher
can move a heavy load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a large load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a fluid load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a load at high speed	◯	◯	◯ ◯ ◯ ◯ ◯
has low maintenance costs	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid wait times at charging stations	◯	◯	◯ ◯ ◯ ◯ ◯
can pick up and drop off goods at different heights	◯	◯	◯ ◯ ◯ ◯ ◯
can process parallel transport orders	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid a standstill	◯	◯	◯ ◯ ◯ ◯ ◯
can recharge fast	◯	◯	◯ ◯ ◯ ◯ ◯

SoS-Zielmodell

Nachfolgend ist das Ihnen **bereits bekannte** SoS-Zielmodell einer Roboterflotte gegeben. Das SoS-Modell zeigt die möglichen Zusammensetzungen der Roboterflotte und das Zielmodell die Ziele der Roboterflotte. Die Beziehungen zwischen den Zusammensetzungen und den Zielen sind ebenfalls dargestellt (Die Farben dienen lediglich zur besseren Unterscheidung und haben keine weitere Bedeutung). Bitte führen Sie jetzt einen Review dieser Spezifikationen durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

The goals/task *pick-ups/drop-offs at different heights* and *move heavy load* can be fulfilled by

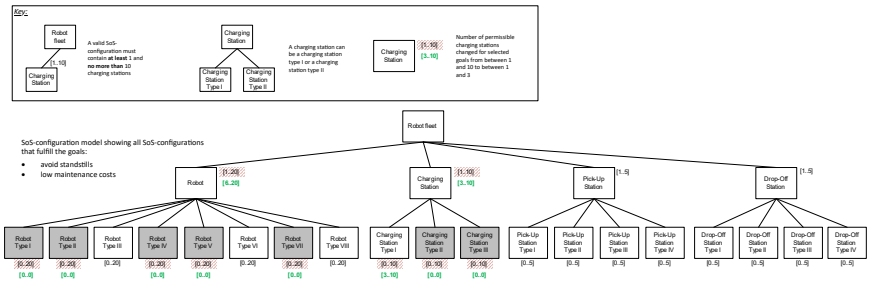
	Dargestellt	Nicht dargestellt	sehr sicher – sehr unsicher
an SoS-configuration consisting of: • four charging stations type I • three robots type III • one robot type VI • three robots type VIII • two pick-up stations type III • one drop-off station type III	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • two charging stations type I • two charging stations type II • four robots type V • one pick-up station type I • one pick-up station type II • one pick-up station type III • one drop-off station type I • one drop-off station type II • one drop-off station type III	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • two charging stations type I • two robots type III • four robots type VI • two robots type VIII • one pick-up station type IV • one drop-off station type III	○	○	○ ○ ○ ○ ○

an SoS-configuration consisting of: • seven charging stations type I • three robots type VI • three robots type VIII • one pick-up station type I • one pick-up station type II • one drop-off station type II • one drop-off station type III	○	○	○○○○○
an SoS-configuration consisting of: • three charging stations type I • three robots of type III • three robots of type VI • two pick-up stations type II • one drop-off station type IV	○	○	○○○○○

Seite 05

SoS-Konfigurationssicht

Nachfolgend ist eine Sicht auf das SoS-Modell einer Roboterflotte gegeben. Diese Sicht stellt die Zusammensetzungen dar, die ausgewählte Ziele erreichen können. Bitte führen Sie jetzt einen Review dieser Sicht durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

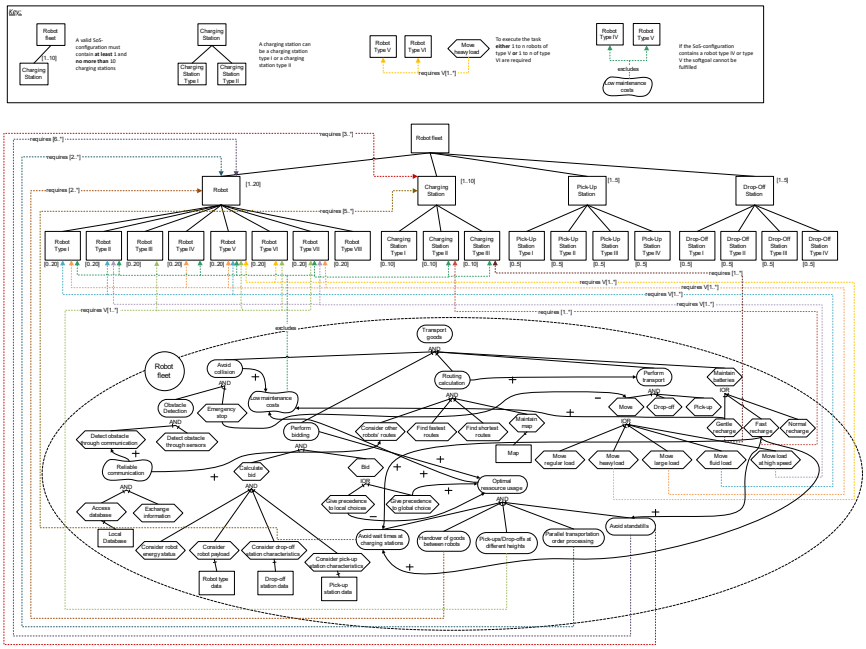
The goals/task *avoid standstills* and *low maintenance costs* can be fulfilled by

	Dargestellt	Nicht dargestellt	sehr sicher – sehr unsicher
an SoS-configuration consisting of: • four charging stations type I • three robots type III • one robot type VI • three robots type VIII • two pick-up stations type III • one drop-off station type III	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • two charging stations type I • two charging stations type II • four robots type V • one pick-up station type I • one pick-up station type II • one pick-up station type III • one drop-off station type I • one drop-off station type II • one drop-off station type III	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • two charging stations type I • two robots type III • four robots type VI • two robots type VIII • one pick-up station type IV • one drop-off station type III	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • seven charging stations type I • three robots type VI • three robots type VIII • one pick-up station type I • one pick-up station type II • one drop-off station type II • one drop-off station type III	○	○	○ ○ ○ ○ ○

an SoS-configuration consisting of: • three charging stations type I • three robots of type III • three robots of type VI • two pick-up stations type II • one drop-off station type IV	○	○	○○○○○

SoS-Zielmodell

Nachfolgend ist ein SoS-Zielmodell einer Roboterflotte gegeben. Das SoS-Modell zeigt die möglichen Zusammensetzungen der Roboterflotte und das Zielmodell die Ziele der Roboterflotte. Die Beziehungen zwischen den Zusammensetzungen und den Zielen sind ebenfalls dargestellt (Die Farben dienen lediglich zur besseren Unterscheidung und haben keine weitere Bedeutung). Bitte führen Sie jetzt einen Review dieser Spezifikationen durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

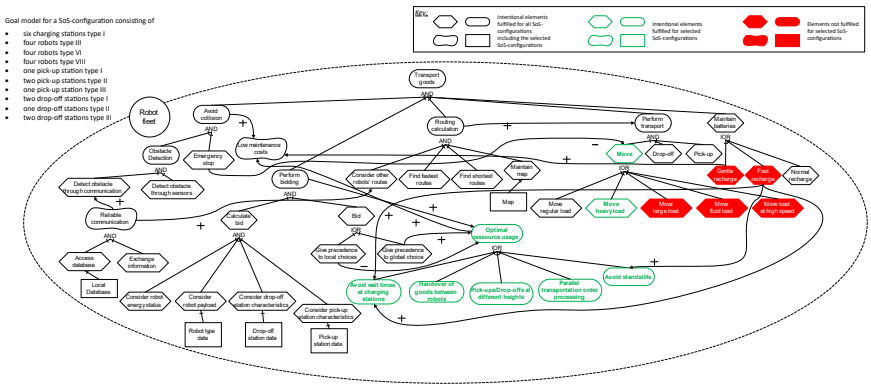
An SoS-configuration consisting of:

- six charging stations type I
- four robots type III
- four robots type VI
- four robots type VIII
- one pick-up station type I
- two pick-up stations type II
- one pick-up station type III
- two drop-off stations type I
- one drop-off station type II
- two drop-off stations type III

	Dargestellt	Nicht dargestellt	sehr sicher ◯ ◯ ◯ ◯ ◯ sehr unsicher
can move a heavy load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a large load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a fluid load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a load at high speed	◯	◯	◯ ◯ ◯ ◯ ◯
has low maintenance costs	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid wait times at charging stations	◯	◯	◯ ◯ ◯ ◯ ◯
can pick up and drop off goods at different heights	◯	◯	◯ ◯ ◯ ◯ ◯
can process parallel transport orders	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid a standstill	◯	◯	◯ ◯ ◯ ◯ ◯

Zielsicht

Nachfolgend ist eine Sicht auf das Zielmodell einer Roboterflotte gegeben. Diese Sicht stellt dar welche Ziele, die Roboterflotte in einer bestimmten Zusammensetzung erreichen kann. Bitte führen Sie jetzt einen Review dieser Sicht durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

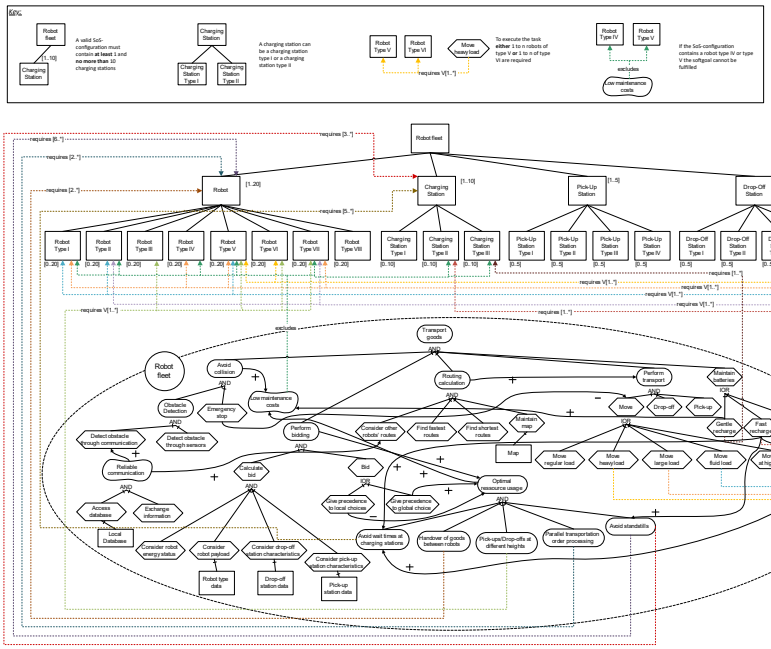
An SoS-configuration consisting of:

- six charging stations type I
- four robots type III
- four robots type VI
- four robots type VIII
- one pick-up station type I
- two pick-up stations type II
- one pick-up station type III
- two drop-off stations type I
- one drop-off station type II
- two drop-off stations type III

	Dargestellt	Nicht dargestellt	sehr sicher ... sehr unsicher
can move a heavy load	○	○	○ ○ ○ ○ ○
can move a large load	○	○	○ ○ ○ ○ ○
can move a fluid load	○	○	○ ○ ○ ○ ○
can move a load at high speed	○	○	○ ○ ○ ○ ○
has low maintenance costs	○	○	○ ○ ○ ○ ○
can avoid wait times at charging stations	○	○	○ ○ ○ ○ ○
can pick up and drop off goods at different heights	○	○	○ ○ ○ ○ ○
can process parallel transport orders	○	○	○ ○ ○ ○ ○
can avoid a standstill	○	○	○ ○ ○ ○ ○

SoS-Zielmodell

Nachfolgend ist das Ihnen **bereits bekannte** SoS-Zielmodell einer Roboterflotte gegeben. Das SoS-Modell zeigt die möglichen Zusammensetzungen der Roboterflotte und das Zielmodell die Ziele der Roboterflotte. Die Beziehungen zwischen den Zusammensetzungen und den Zielen sind ebenfalls dargestellt (Die Farben dienen lediglich zur besseren Unterscheidung und haben keine weitere Bedeutung). Bitte führen Sie jetzt einen Review dieser Spezifikationen durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

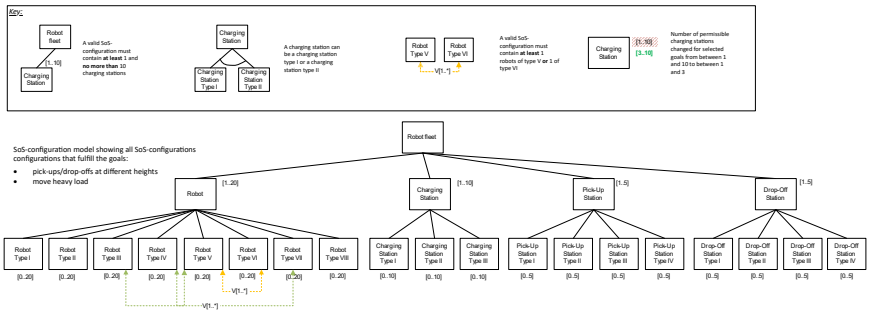
An SoS-configuration consisting of:

- one charging station type I
- one charging station type III
- three robots type II
- three robots type V
- one robot type VIII
- two pick-up stations type II
- two drop-off stations of type II

	Dargestellt	Nicht dargestellt	sehr sicher ◯ ◯ ◯ ◯ ◯ sehr unsicher
can move a heavy load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a large load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a fluid load	◯	◯	◯ ◯ ◯ ◯ ◯
can move a load at high speed	◯	◯	◯ ◯ ◯ ◯ ◯
has low maintenance costs	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid wait times at charging stations	◯	◯	◯ ◯ ◯ ◯ ◯
can pick up and drop off goods at different heights	◯	◯	◯ ◯ ◯ ◯ ◯
can process parallel transport orders	◯	◯	◯ ◯ ◯ ◯ ◯
can avoid a standstill	◯	◯	◯ ◯ ◯ ◯ ◯
can recharge fast	◯	◯	◯ ◯ ◯ ◯ ◯

SoS-Konfigurationssicht

Nachfolgend ist eine Sicht auf das SoS-Modell einer Roboterflotte gegeben. Diese Sicht stellt die Zusammensetzungen dar, die ausgewählte Ziele erreichen können. Bitte führen Sie jetzt einen Review dieser Sicht durch.



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence“ an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

The goals/task *pick-ups/drop-offs at different heights and move heavy load can be fulfilled by*

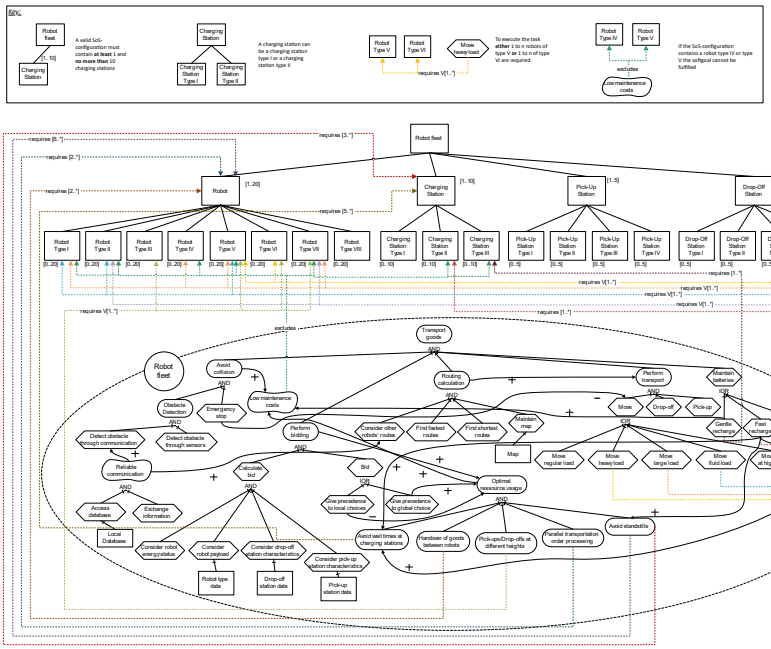
	Dargestellt	*Nicht dargestellt*	*sehr sicher* — *sehr unsicher*
an SoS-configuration consisting of: • *four charging stations type I* • *three robots type III* • *one robot type VI* • *three robots type VIII* • *two pick-up stations type III* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *two charging stations type I* • *two charging stations type II* • *four robots type V* • *one pick-up station type I* • *one pick-up station type II* • *one pick-up station type III* • *one drop-off station type I* • *one drop-off station type II* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *two charging stations type I* • *two robots type III* • *four robots type VI* • *two robots type VIII* • *one pick-up station type IV* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *seven charging stations type I* • *three robots type VI* • *three robots type VIII* • *one pick-up station type I* • *one pick-up station type II* • *one drop-off station type II* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○

an SoS-configuration consisting of: • *three charging stations type I* • *three robots of type III* • *three robots of type VI* • *two pick-up stations type II* • *one drop-off station type IV*	○	○	○○○○○

Seite 10

SoS-Zielmodell

*Nachfolgend ist das Ihnen **bereits bekannte** SoS-Zielmodell einer Roboterflotte gegeben. Das SoS-Modell zeigt die möglichen Zusammensetzungen der Roboterflotte und das Zielmodell die Ziele der Roboterflotte. Die Beziehungen zwischen den Zusammensetzungen und den Zielen sind ebenfalls dargestellt (Die Farben dienen lediglich zur besseren Unterscheidung und haben keine weitere Bedeutung). Bitte führen Sie jetzt einen Review dieser Spezifikationen durch.*



Sind die nachfolgenden Aussagen dargestellt?

Bitte geben Sie auch Ihre „Confidence" an. D.h. Ihre eigene Zuversicht, dass die von Ihnen getroffene Entscheidung richtig ist.

The goals/task avoid standstills and low maintenance costs can be fulfilled by

	Dargestellt	*Nicht dargestellt*	*sehr sicher* ... *sehr unsicher*
an SoS-configuration consisting of: • *four charging stations type I* • *three robots type III* • *one robot type VI* • *three robots type VIII* • *two pick-up stations type III* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *two charging stations type I* • *two charging stations type II* • *four robots type V* • *one pick-up station type I* • *one pick-up station type II* • *one pick-up station type III* • *one drop-off station type I* • *one drop-off station type II* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *two charging stations type I* • *two robots type III* • *four robots type VI* • *two robots type VIII* • *one pick-up station type IV* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○
an SoS-configuration consisting of: • *seven charging stations type I* • *three robots type VI* • *three robots type VIII* • *one pick-up station type I* • *one pick-up station type II* • *one drop-off station type II* • *one drop-off station type III*	○	○	○ ○ ○ ○ ○

an SoS-configuration consisting of: • *three charging stations type I* • *three robots of type III* • *three robots of type VI* • *two pick-up stations type II* • *one drop-off station type IV*	○	○	○○○○○

Seite 11

Bitte bewerten Sie Ihr Empfinden bei den beiden von Ihnen durchgeführten Reviews

Welches Modell würden Sie bevorzugen?					
	das SoS-Zielmodell				*die Sichten*
Meine Performance im manuellen Review wird am besten durch folgendes Modell unterstützt	○	○	○	○	○
Meine Produktivität im manuellen Review wird am besten durch folgendes Modell unterstützt	○	○	○	○	○
Ich bin am effektivsten wenn ich folgendes Modell für den manuellen Review nutze	○	○	○	○	○
Ich finde folgendes Modell am nützlichsten, um einen manuellen Review durchzuführen	○	○	○	○	○
Für mich ist die Benutzung folgenden Modells im manuellen Review am einfachsten und verständlichsten	○	○	○	○	○
Der Review folgenden Modells ist für mich mit den wenigsten mentalen Anstrengungen verbunden	○	○	○	○	○
In manuellen Reviews finde ich folgendes Modell einfacher zu nutzen	○	○	○	○	○
Wenn es niemanden gäbe, der mir eine Einführung geben könnte, würde ich folgendes Modell für den manuellen Review bevorzugen	○	○	○	○	○
Wenn es jemanden gäbe, der mir zu Beginn zeigen würde, wie ich das Modell nutzen kann, würde ich folgendes Modell für den manuellen Review bevorzugen	○	○	○	○	○
Wenn ich schon vorher vergleichbare Modelltypen genutzt hätte, würde ich den manuellen Review folgenden Modells bevorzugen	○	○	○	○	○

Seite 12

Erfahrung und Biografische Angaben

Bitte schätzen Sie Ihre Erfahrung mit den nachfolgenden Entwicklungsaktivitäten und Notationsformen ein.					
	sehr erfahren				*sehr unerfahren*
Modellbasiertes Requirements Engineering	○	○	○	○	○
Entwicklung eingebetteter Systeme	○	○	○	○	○
Entwicklung variabler Systeme	○	○	○	○	○
Zielmodelle	○	○	○	○	○
Feature Modelle	○	○	○	○	○
Der Review modellbasierter Spezifikationen	○	○	○	○	○

Bitte unterstützen Sie uns mit einigen biografischen Informationen.	
Alter	*Jahre*
Geschlecht	[Bitte auswählen]
Höchster bisheriger Bildungsabschluss (falls zutreffend mit Studiengangsname)	
Aktuelle Haupttätigkeit	[Bitte auswählen]

Seite 13

Bitte unterstützen Sie uns noch mit einigen weiteren biografischen Daten. Die Daten werden selbstverständlich vollständig anonym ausgewertet.

question('PD02')

Angestrebter Studienabschluss	
Studiengang	
seit x Semestern in diesem Studiengang eingeschrieben	*Semester*
Vorraussichtliches Studienende	

question('PD03')

Berufsbezeichnung	
In aktueller Stellung seit	*Jahre*
Berufserfahrung in Jahren	*Jahre*