Fachhochschule Aachen

Fachbereich Elektrotechnik und Informationstechnik

Heinrichs, Marius

Bachelorarbeit

Zur Erlangung des akademischen Grades

Bachelor-Informatiker (FH)

Wie individuell gestaltbare Plugins den Model-Based Systems Engineering Prozess unterstützen können.

Eingereicht von:

Heinrichs, Marius

Matrikelnummer: 3217081

Studienrichtung: Informatik

31.05.2023

Betreuer: Voss, Sebastian, Prof. Dr. rer. nat.

Koreferent: Höner, Tim, M. Eng.

Kurzfassung

Diese Arbeit hat das Ziel, herauszufinden mit welchen Kriterien sich Strukturdiagramme aus der SysML Bewerten lassen und wie diese Bewertung automatisiert werden kann. Um dem auf den Grund zu gehen, wird die folgende Forschungsfrage gestellt: Wie lässt sich die Qualität von Strukturdiagrammen der SysML, in Bezug auf die Semantik und Syntax, bewerten und wie kann diese Bewertung automatisiert werden kann?

Um diese Frage beantworten zu können, wurde eine qualitative Forschung betrieben, mit dem Ziel, Bewertungskriterien aus Empfehlungen, Good Practices und Prinzipien, vorgestellt von Experten auf ihrem Gebiet, abzuleiten. Die Bewertung der Kriterien wurde dann, über die MagicDraw API für den Cameo Systems Modeler, implementiert.

Durch die erhobenen Bewertungskriterien lässt sich schlussfolgern, dass Strukturdiagramme Bewertbar sind, jedoch nicht alle Kriterien sich für eine automatisierte Bewertung eignen. Weiter kann diese Arbeit als eine Quelle für verschiedenen Modellierungsprinzipen genutzt werden. Zusätzlich konnte diese Arbeit eine Wissenschaftliche Basis schaffen, die Aussage tätigt, was qualitative Eigenschaften der Strukturdiagramme sind.

Diese Arbeit ist auch an Grenzen gestoßen und Empfehlungen für weiterführende Forschung wurden erkennbar. So können mehr Bewertungskriterien erhoben werden, das Plugin um zusätzliche Funktionalitäten erweitert werden und sogar eine automatisierte Nutzwertanalyse mit Graphischem Nutzer Interface kann implementiert werden. Zusätzlich bietet es sich an Forschung zu betreiben, um die Namenskonventionen und Diagrammstrukturierung zu standardisieren.

Schlagwörter:

BDD, IBD, PKG, Strukturdiagramme, Good Practices, Heuristiken, SysML, Systems Engineering, Model-Based Systems Engineering, Software-Engineering, Qualitätsbewertung, Automatisierung, Nutzwertanalyse, warning, smells, errors, semantisch, syntaktisch

Inhaltsverzeichnis

Kurzfassung II

Abbildungsverzeichnis V

Tabellenverzeichnis VI

Vorwort VII

1 Einleitung 1

1.1 Thema und Forschungsfrage 1

1.2 Forschungsstand und Relevanz 1

1.3 Das Problem und Ziel 3

1.4 Wie geht es zum Ziel? 3

1.5 Aufbau der Bachelorarbeit 4

2 Theoretischer Rahmen 5

2.1 Definition der Schlüsselbegriffe 5

2.2 Präsentation relevanter Prinzipien, Theorien und Konzepte 6

3 Methodik 9

3.1 Art und Weise der Forschung 9

3.2 Die Daten und ihre Analyse 9

3.3 Die erhobenen Bewertungskriterien 10

3.3.1 Bewertungskriterien für Block Definition Diagramme 10

3.3.2 Bewertungskriterien für Interne Block Diagramme 12

3.3.3 Bewertungskriterien für Packet Diagramme 14

3.4 Gütekriterien 15

4 Ein Plugin für die automatisierte Einstufung des Verletzungsgrades von Bewertungskriterien 16

4.1 Modellierung eines minderwertigen BDD's und die Bewertung des Plugins 17

4.2 Modellierung eines minderwertigen IBDs und die Bewertung des Plugins 27

5 Zusammenfassung und Ausblick 34

5.1 Zusammenfassung der Ergebnisse 34

5.2 Interpretation der Ergebnisse 37

5.3 Begrenzungen dieser Forschung 38

5.4 Empfehlungen für weiterführende Forschungen 38

6 Fazit 40

Glossar VIII
Literaturverzeichnis X
Anhänge XIV
Erklärung XV

Abbildungsverzeichnis

Abbildung 1: Blöcke starten mit einem Lower Case 17

Abbildung 2: Nicht jeder Port eines Blockes ist typisiert 18

Abbildung 3: Nicht jeder Port ist ein ProxyPort, verschiedene Typen von Ports wurden benutzt. 19

Abbildung 4: Zu viele Input-, Output-Ports. Nicht jeder Port besitzt eine Richtung 21

Abbildung 5: Dekomposition ist nicht korrekt 23

Abbildung 6: Die Tiefe der Kompositions Assoziation fängt an zu groß zu werden. 25

Abbildung 7: Das SuD besitzt kein IBD 26

Abbildung 8: IBD des WorstCase SuD 27

Abbildung 9: Part Property im IBD startet mit einem UpperCamelCase 28

Abbildung 10: Nicht jeder Port im IBD ist typisiert und besitzt ein Flow Property. Zusätzlich fehlt eine Connector Line 29

Abbildung 11: Zu viele Input-, Output-Ports am Part. 30

Abbildung 12: Es sind nicht nur Proxy Ports vorhanden, es wurde in einem IBD verschiedene Arten von Ports verwendet. 31

Abbildung 13: Parts Kommunizieren mit nicht direkten Nachbarn 33

Tabellenverzeichnis

Tabelle 1: Bewertungskriterien für Block Definition Diagramme.....35

Tabelle 2: Bewertungskriterien für Interne Block Diagramme 35

Tabelle 3: Bewertungskriterien für Paket Diagramme.....36

Tabelle 4: Funktionen des Plugins.....37

Vorwort

Während meinem Studium habe ich ein Interesse für das Model-Based Systems Engineering entwickelt. Spezieller, den Umgang mit Cameo Systems Modeler und der Modellierung von Systemen mit Hilfe der Systems Modelling Language.

Das Praxisprojekt an der Fachhochschule Aachen in dem Fachbereich 5 gab mir den Anstoß mich tiefer mit dem Thema zu befassen und mir zur Aufgabe zu machen, Bewertungskriterien für eine Qualitative Nutzwertanalyse zu erheben und mit der Entwicklung eines Plugins die Bewertung von Diagrammen zu automatisieren.

An dieser Stelle möchte ich mich noch bei Prof. Dr. rer. Nat. Voss, für das Angebot ein Praxisprojekt und die anschließende Bachelorarbeit mit ihm als Betreuer durchzuführen, bedanken. Weiterer Dank gilt meinen Kommilitonen, die an ähnlichen Praxisprojekten beteiligt waren, mit denen ich mich bei der Entwicklung mit der MagicDraw API austauschen konnte.

1 Einleitung

1.1 Thema und Forschungsfrage

Das Thema dieser Arbeit ist wie individuell gestaltbare Plugins den Model-Based Systems Engineering Prozess unterstützen.

Das Systems Engineering ist ein sehr komplexes und großes Themengebiet, und lässt sich nicht nur in der Informatik wieder finden. Den ersten bedeutenden Einsatz, hatte eine Systems Engineering Methodik im Jahr 1940 (Geschichte - Systems Engineering, 2023). Systems Engineering ist ein Fachübergreifender und integrierender Ansatz der, die erfolgreiche Realisation, nutzen und die Pensionierung von entwickelten Systemen ermöglicht. Dafür nutzt das Systems Engineering System Prinzipien und Konzepte, außerdem wissenschaftliche, technologische und Management Methodiken (International Council on Systems Engineering (INCOSE), 2023).

Seitdem sind viele verschiedene Systems Engineering Methodiken entstanden und haben sich bis heute etabliert, wobei sich andere noch in der Entstehungsphase befinden (International Council on Systems Engineering (INCOSE), 2022).

Das Model-Based Systems Engineering, als eine der entstandenen Methodiken, bildet Informationen über ein zu entwickelndes System nicht nur über textbasierte Dokumente ab, sondern vor allem über Modelle, strukturiert mit Diagrammen. Model-Based Systems Engineering ist die formalisierte Anwendung des Modellierens, um System Anforderungen, Design, Analyse, Verifizierung und Validierung zu unterstützen. All das geschieht durch allen „Lifecycle“ Phasen hinweg (International Council on Systems Engineering (INCOSE), 2007).

Diese wissenschaftliche Arbeit beschäftigt sich spezieller mit der Forschungsfrage, wie sich die Qualität von Strukturdiagrammen der SysML (Systems Modeling Language), in Bezug auf die Semantik und Syntax, bewerten lässt und wie diese Bewertung automatisiert werden kann.

1.2 Forschungsstand und Relevanz

Die Model-Based Systems Engineering Methodik steckt in seiner Entwicklung noch in einer sogenannten Übergangsphase. (International Council on Systems Engineering (INCOSE), 2022) Es fehlt eine breite Theoretische Basis, die auf wissenschaftliche und mathematische Grundlagen aufbaut. Zudem sind System Architektur Methoden, zum Beispiel für das Bewerten nach Qualitätskriterien einer solchen Architektur, nicht genügend etabliert (International Council on Systems Engineering (INCOSE), 2022).

Jedoch findet MBSE (Model-Based Systems Engineering) immer mehr Anklang in der Systems Engineering Community und wird von vielen als die Zukunft für das Systems Engineering angesehen (International Council on Systems Engineering (INCOSE), 2022). Das

Bedeutet auch, dass immer mehr Forschung auf diesem Themengebiet betrieben wird und einiges an etablierter Literatur und erfahrenen Anwendern vorhanden sind.

Ein Paper von Towers James und Hazle Alex hat bereits „Good Practices“ in Model Verifikation und Validation untersucht. Sie haben eine Liste mit Techniken und Prinzipien erstellt, die wenn eingesetzt einen effektiven und robusten Ansatz für Modell Verifikation und Validation bieten. So sollten Modellierungsmuster genutzt werden und ein Modellierungsstandard definiert werden. Die Analyse sollte zusätzlich automatisch die Syntax des Modells überprüfen (Towers & Hazle, 2020).

„Cookbook for MBSE with SysML“ ist ein Technischer Report mit dem Ziel Modellierungsmuster, Rezepte, Richtlinien und Best Practices für die Anwendung der SysML zu Verfügung zu stellen (Hein, Weilkiens, Karban, & Zamparelli, 2011).

Wie gerade bewiesen, gibt es zwar einige Quellen für Modellierungsprinzipien, allerdings so gut wie keine Arbeit bezüglich der Automatisierung dieser. Obwohl die Automatisierung von vielen empfohlen wird.

Der Trend im Nutzen von Model-Based Systems Engineering Methodiken stieg in den letzten 9 Jahren konstant an. Model-Based Ansätze finden sich nicht mehr nur zum Großteil in der Verteidigungs- und Raumfahrtindustrie, sondern auch in der Luftfahrt, Automobilindustrie, Informatik und in der Medizin. Andere Industrien wie die Transportations- oder die Energieindustrie finden auch immer mehr ihren Weg zu einem Model-basierten Ansatz. System Ingenieure, System Ingenieur Management und die Kunden finden Mehrwert im Nutzen von Modellen. Allerdings gibt es immer noch einen Mangel an Können, entstehend aus dem Mangel an Trainings Angeboten von Organisationen/Firmen (Systems Engineering Body of Knowledge (SEBoK), 2023). Dieser Mangel kann mit Hilfe einer automatisierten Qualitätsanalyse der Diagramme ausgeglichen werden.

Zudem werden Systeme zunehmend komplexer und müssen immer mehr Anforderungen genügen. Auch die inter-Konnektivität von Systemen wird zahlreicher und die Anfälligkeit und das Risiko steigt (International Council on Systems Engineering (INCOSE), 2022). Ein gutes Diagramm verringert die Anfälligkeit und minimiert das Risiko. Ein gutes Diagramm wird, durch die Einhaltung der Eigenschaften, wie der Übersicht, geschaffen.

Das Entwickeln und Modellieren von qualitativ hochwertigen Systemen wird immer wichtiger um die Sustainable Development Goals (SDGs), vorgestellt von den Vereinigten Staaten von Amerika, einzuhalten. Aus diesem Grund entschied sich die US National Academy of Engineering (NAE) dazu Grand Challenges für das Engineering festzulegen. Darunter zählen unteranderem: Develop carbon sequestration methods, Restore, and improve urban Infrastructure, Advance health informatics, Reverse-Engineer the brain und 10 weitere Herausforderungen (National Academy of Engineering, 2023).

Meine persönliche Motivation eine Wissenschaftliche Arbeit über das Automatisieren der Qualitätsbewertung von Struktur Diagrammen zu verfassen, ist ein Blick in die Zukunft. Systems Engineering wird immer wichtiger, nicht nur in der Informatik werden Systeme

gebraucht, auch der normale Bürger profitiert von vielen Systemen wie den Gesundheitssystemen.

Des Weiteren ist ein Mangel an ausgebildeten System Ingenieuren ersichtlich und eine große Nachfrage auf dem Arbeitsmarkt wird vorhanden sein. Ein Trend in der Industrie zeigt einen Trend in sogenannte „Smart-Factories“, bei denen Produkte überwacht und deren stand in der Produktion und Versendung bekannt sind. Außerdem sind deren Hardware und Softwarekonfigurationen, innerhalb jeden Schrittes ihres Lifecycles, individuell katalogisiert. Das Bedeutet die Industrie ist gerade dabei sich neu zu erfinden. Diese Entwicklung wurde von der Deutschen Akademie der Technikwissenschaften (acatech) unter den Begriff „Industrie 4.0“ geprägt (International Council on Systems Engineering (INCOSE), 2022).

„Der Begriff „Industrie 4.0“ steht seit 2011 für die massenhafte Verbindung von Informations- und Kommunikationstechnologien mit der industriellen Produktion. Mit Industrie 4.0 können Prozesse, Routinen und Systeme effizienter gestaltet, aber auch neue Möglichkeiten zur Differenzierung des Leistungsangebots geschaffen werden.“ (Deutsche Akademie der Technikwissenschaften (acatech), 2020) Ein klarer Anwendungsfall für Systems Engineering und daraus folgend für eine Modell basierte Methodik. Gerade die Effizienz in der Gestaltung von Systemen, profitiert von einer automatisierten Qualitätsanalyse.

1.3 Das Problem und Ziel

Model-Based Systems Engineering ist eine relative junge Methodik, dementsprechend sind Informationen über viele verschiedene, meist unter den übergeordneten Begriff des Systems Engineering versteckt, Fachliteraturen verstreut. Zwar gibt es schon einige Versuche das Wissen über Model-Based Ansätze zu bündeln, wie die SEBok Wiki, doch erstaunlich wenig bemühen Qualitätsmerkmale von Diagrammen zusammenzutragen und genau zu bestimmen.

Das Ziel dieser Forschung ist es bereits etablierte „Heuristiken“, die dabei helfen ein Modell syntaktisch und semantisch qualitativ hochwertig zu modellieren, zusammenzutragen und daraus Bewertungskriterien zu bilden. Die Bewertungskriterien sollen Aussage über die Qualität eines Diagrammes tätigen und somit dem System Ingenieur in seinem Modellierungsprozess unterstützen. Als eine beispielhafte Bewertung von Qualitätskriterien auf Strukturdiagramme wurde, für das Cameo Systems Modeler Werkzeug, ein Plugin (siehe Anhang 2: Plugin Source Code) implementiert und angewendet.

Dadurch soll ein Anstoß, in das Bilden einer global einheitlich genutzten Quelle für Modellierungsprinzipien mit SysML, gegeben werden. Hauptsächlich aber soll ein Grundstein gelegt werden, die Bewertung der Qualität von Diagrammen zu automatisieren.

1.4 Wie geht es zum Ziel?

Für die Beantwortung der Forschungsfrage bedient sich diese Arbeit einer qualitativen Forschungsmethode, mit dessen Hilfe Bewertungskriterien erhoben werden. Mit diesen Daten

wird dann das Automatisieren einer Nutzwertanalyse auf Strukturdiagramme ermöglicht und als Plugin für den Cameo Systems Modeler implementiert.

Eine qualitative Forschungsmethodik wurde gewählt, um ein möglichst breites Abbild von Bewertungskriterien zu erzeugen, analysieren und zu interpretieren. Mit den qualitativ gesammelten Kriterien und der Hilfe einer Nutzwertanalyse lässt sich dann die Qualität, oder den Nutzen, von Strukturdiagrammen bewerten.

Dafür wurden Empfehlungen von Experten auf ihrem Gebiet und etablierte Prinzipien gesammelt und analysiert. Das Ziel der Analyse ist es Bewertungskriterien abzuleiten und diese Softwaretechnisch zu implementieren.

1.5 Aufbau der Bachelorarbeit

Die Arbeit ist auf folgender Weise strukturiert:

Zuerst werden die wichtigsten Schlüsselbegriffe definiert und erklärt, um eine gemeinsame Basis zu schaffen. Im Anschluss dessen werden Relevante Theorien, Prinzipien und Konzepte vorgestellt, die dabei helfen werden, die gebildeten Bewertungskriterien zu begründen.

Im nächsten Teil wird die Methodik beziehungsweise die Vorgehensweise bei der Erhebung der Daten und der Verarbeitung erläutert. Die erhobenen Good Practices, Heuristiken, und Prinzipien werden dann analysiert und zu Bewertungskriterien gewandelt.

Als nächstes wird das Plugin vorgestellt und mit Hilfe von Abbildungen und Codeausschnitten verdeutlicht, wie das Plugin die automatisierte Überprüfung umsetzt.

In der Zusammenfassung werden alle Bewertungskriterien und Funktionen des Plugins tabellarisch festgehalten. Die Ergebnisse werden dann interpretiert und Schlussfolgerungen werden getroffen. Grenzen, auf die diese Arbeit gestoßen ist, werden kurz besprochen und im gleichen Zuge Empfehlungen für weiterführende Forschung gegeben.

Zum Schluss wird im Fazit die Forschungsfrage beantwortet. Die wichtigsten Ergebnisse und die Empfehlungen für zukünftige Forschung zusammengefasst.

2 Theoretischer Rahmen

In diesem Teil der Arbeit werden zuerst einige Grundlagen aufgearbeitet. Im Anschluss werden relevante Theorien und Konzepte vorgestellt.

2.1 Definition der Schlüsselbegriffe

Da sich diese Arbeit mit dem Bewerten von bestimmten Diagrammen in der SysML, bezogen auf die Syntax und der Semantik befasst, müssen zuerst einige Konzepte erläutert werden. Diese Konzepte bilden dann die Grundlage, um zu verstehen, was mit Semantik und Syntax bei der Modellierung von Diagrammen gemeint ist.

Diagramme sind eine grafische Darstellung von Daten, Sachverhalten oder Informationen. SysML nutzt diese Diagramme, um einen Ausschnitt des Models grafisch darzustellen.

Ein Modell wiederum, ist eine vereinfachte Repräsentation eines Systems zu einem bestimmten Zeitpunkt, mit dem Ziel einen tieferen Blick in die verschiedenen System Aspekte, wie Funktionen, Struktur, Eigenschaften, Leistung, Verhalten oder Kosten, zu erhalten (Friedenthal, Dori, & Mordecai, Representing_Systems_with_Models, 2023).

Der Begriff „modellieren“ bedeutet, etwas zu gestalten, zu bilden und zu formen. Speziell im Bereich der Informatik wird der Begriff oft genutzt, wenn man ein Ding der Realität auf die wesentlichen Elemente runterbricht und textuell oder grafisch abbilden will. Diese Abbildung der Realität nennt man auch Modell. So ein Modell kann mittels Diagramme aus der SysML gebildet oder anders gesagt modelliert werden. Der Modellierung Prozess wird auch Systems Modelling genannt und ist eine der grundlegendsten Systems Engineering Vorgänge (International Council on Systems Engineering (INCOSE), 2022).

Die SysML ist eine graphische Modellierungssprache zum Spezifizieren, Analysieren, Designen und Verifizieren von komplexen Systemen. Sie bietet eine graphische Repräsentation mit einer semantischen Grundlage für das Modellieren von System Anforderungen, Verhalten und Parametrik (Object Management Group, 2023).

Die Systems Modelling Language besteht aus insgesamt neun verschiedenen Diagramm Typen und einem Allocation Table. Diese Arbeit befasst sich genauer mit den sogenannten Struktur Diagrammen, darunter befindet sich das Block-Definitions-Diagramm, das Interne-Block-Diagramm, das Packet-Diagramm und dem Parametrik-Diagramm. Das Parametrik-Diagramm wird allerdings nicht weiter betrachtet.

Ein Block-Definitions-Diagramm ist ein statisches Strukturdiagramm, welches die Systemkomponenten und deren Eigenschaften, Verhalten, Limitierungen, funktionalen Schnittstellen und Beziehungen abbildet (SysML, 2023). Manchmal auch die „Black Box“ Sicht auf das System genannt.

Das Interne-Block Diagramm repräsentiert dementsprechend die sogenannte „White Box“ Sicht. Es handelt sich um ein weiteres Struktur Diagramm, dass einen bestimmen Block zugehörig ist und dessen Struktur widerspiegelt. Darunter zählen die zugehörigen Einzelteile,

Eigenschaften, Konnektoren, internen Schnittstellen und Funktionellen Schnittstellen (SysML, 2023).

Das dritte und letzte Strukturdiagramm, welches in Betracht gezogen wird, ist das Paket-Diagramm. Das Paket-Diagramm zeigt den Zusammenhang zwischen Paketen und deren Inhalt. Ein Paket ist ein Mechanismus, für das Organisieren von Elementen und Diagrammen. Pakete definieren zusätzlich einen einzigartigen „Namespace“ für Modell Elemente (SysML, 2023).

Die Syntax einer Sprache beschreibt Regeln, nach denen sprach Konstrukte gebildet werden. Im Falle der SysML, sind das Regeln nach denen Diagramme modelliert werden. Die Semantik einer Sprache beschreibt die Bedeutung der sprach Konstrukte. Die Semantik in Diagrammen ist die Bedeutung, die den Diagramm Elementen gegeben wird und somit auch des Diagramms an sich.

Wird die Semantik oder Syntax im Modellierungsprozess verletzt, besteht die Gefahr auf eine Qualitätsminderung der Diagramme. Eine Qualitätsminderung ist dann vorhanden, wenn die Übersichtlichkeit oder Flexibilität des Diagramms verloren geht, das Diagramm fehleranfälliger wird, Das Diagramm keine oder eine falsche Semantik besitzt, Informationen verloren gehen, Missverständnisse entstehen oder Verwechslungsgefahr besteht.

Um dem entgegenzuwirken, werden Good Practices, Empfehlungen und Heuristiken genutzt.

2.2 Präsentation relevanter Prinzipien, Theorien und Konzepte

Da nun die Grundlagen geklärt sind, können einige Prinzipien und Konzepte erläutert werden. Diese helfen dabei, die aus den gesammelten Heuristiken gebildeten Bewertungskriterien auf vorhandene Konzepte zurückzuführen und zu belegen.

Eines dieser wichtigen Prinzipien ist das KISS-Prinzip, KISS steht für „Keep it simple stupid“ oder auch „Keep it short and simple“. Hierbei handelt es sich um ein Designprinzip, welches besagt, dass die meisten Systeme am besten Arbeiten, wenn sie simpel anstatt kompliziert gehalten werden, weswegen Unkompliziertheit eine Schlüsselrolle im Design spielt. Der Ursprung des Satzes “Keep it short and simple”, wird meist Clarence Kelly Johnson zugeordnet (Rich, 1995), ein ehemaliger leitender Ingenieur, der bei Lockheed Skunk Works gearbeitet hat. Der Ursprung des Konzeptes ist aber auf den Zeitungsartikel der Minneapolis Star im Jahr 1938 zurückzuführen, dort wurde das Konzept unter der Phrase: „Keep it short and simple“ definiert (Keep It Short and Simple, 1938).

Ein weiteres Konzept aus der objektorientierten Programmierung ist das Gesetz von Demeter. Das Gesetz wurde erstmals 1988, in der Form einer wissenschaftlichen Veröffentlichung, in der 23. Ausgabe des Buches mit dem Titel „oopsla“ vorgestellt. Das Gesetzt verspricht bei Befolgung, einfachere Instandhaltung von Software, weniger Beziehungen zwischen Methoden, besseres verstecken von Informationen, kleinere Interfaces, einfacheres wiederverwenden von Methoden und zuletzt eine einfachere Prüfung auf Richtigkeit. Zeitgleich wird Redundanz entfernt, die Anzahl der Argumente von Methoden und die Menge an Methoden verringert (Lieberherr, Holland, & Riel, 1988). Im Wesentlichen geht es darum, dass

Objekte nur mit Objekten in ihrer unmittelbaren Umgebung interagieren sollen und dadurch eine Entkopplung von Softwaresystemen stattfindet.

Dem Bereich der Modellbildung entstammt eines der wichtigsten Konzepte, für das Bewerten von Struktur Diagrammen und ihrer Qualität, die Dekomposition. Die Anwendung einer Dekomposition ist das Methodische Zerlegen einer Struktur und sollte bei Struktur Diagrammen in der SysML nach gewissen Regeln erfolgen. Bei Missachtung der Regeln, ist die Syntax des dekomponierten Systems verletzt und unter Umständen ist auch die Semantik davon betroffen.

Das Buch „Clean Code: A Handbook of Agile Software Craftsmanship“ erschien im Jahr 2008 und beschäftigt sich mit der Beantwortung der Frage, was gutes Programmieren ist, wie man gut programmiert und was Guten von schlechtem Code unterscheidet. Es ist eines der meist empfohlenen Literaturen in der Agilen Softwareentwicklung und wurde von Robert C. „Uncle Bob“ Martin geschrieben. R. Martin arbeitet seit den 1970 in der Softwareentwicklung und ist fundamental an der Entwicklung der agilen Softwareentwicklung Methodik beteiligt gewesen (Highsmith, 2023). Einige seiner Erkenntnisse aus dem Buch Clean Code, lassen sich auch auf das Model-Based Systems Engineering übertragen und dienen als Kriterien für das Bestimmen der Qualität eines Diagrammes.

Webel IT Australia ist eine Wissenschaftliche IT-Beratungsstelle, die sich unteranderem auf das Model-Based Systems Engineering mit der grafischen Systems Modeling Language und der Softwarearchitektur und dem Softwareengineering mit der Unified Modeling Language spezialisiert hat. Webel ist seit 2000 in Betrieb und bietet IT-Training für die SysML und UML (Unified Modeling Language) an. Der Lehrer dieser Kurse ist Dr. Darren R. C. Kelly, welcher Webel im Jahr 2000 gegründet hat, um die Applikation von modernen IT-Methoden in der Industrie, Engineering, Handel, Wissenschaft und in der Bildung zu bewerben (Darren, Model-Based Systems Engineering and Software Engineering resume, 2016). Er verfasste die “Webel’s Best Practice policy notes for UML and SysML1.x and the MagicDraw/Cameo tools” welche als der umfassendste Leitfaden für das robuste, flüssigste, angenehmste und konsistenteste modellieren für MBSE beworben wird (Darren, Webel's Best Practice policy notes for UML and SysML1.x and the MagicDraw/Cameo tools, 2023). Viele seiner Erkenntnisse lassen sich als Maß für die Qualität von Diagrammen verwenden.

Eine weitere Quelle, die über SysML und ihre korrekte Anwendung berichtet, ist das Buch „A Practical Guide to SysML: The Systems Modeling Language“ von Sanford Friedenthal, Alan Moore und Rick Steiner. Die Erstauflage erschien am 24. Juli 2008 und ist in vier Abschnitte unterteilt, darunter befindet sich die Einleitung, die Sprach Beschreibung, Beispiele von MBSE-Methoden und zuletzt der Umstieg in das Model-Based Systems Engineering (Friedenthal, Moore, & Steiner, Book Organization, 2015). Das Buch repräsentiert einen Praxisleitfaden für ein breites Spektrum von Anwendern der SysML in der Industrie und dem Studium. Es kann als Einstieg und Referenz für Fachkräfte dienen. Besonders relevant für die Bewertung der Qualität von Diagrammen, ist das Kapitel 2.2.4 „Establishing Model Quality Criteria“. Die vorgestellten Modell Qualitäts Kriterien können genutzt werden, um zu bestimmen, wie gut ein Modell seinen vorgesehenen Nutzen erfüllt. Allerdings gibt es einen Unterschied zwischen

einem guten Modell und einem guten Design. Ein Design kann bewertet werden, je nachdem wie gut es seine Anforderungen erfüllt und in welchem Ausmaß qualitative design Prinzipien angewandt wurden (Friedenthal, Moore, & Steiner, Establishing Model Quality Criteria, 2015).

Die Webseite SysML.org beinhaltet Informationen über die SysML Partners und deren“ SysML Open Source Specification Project“, welches den Systems Modeling Language Dialekt für die Unified Modeling Language v. 2 für Systems Engineering Anwendungen im Jahre 2003 erschaffen hat (SysML Partners, 2023). Auf deren Webseite werden nicht nur die einzelnen Sprach Konstrukte vorgestellt, sondern auch einige „Best Practice Patterns“ und „Anti Patterns“ werden für die Modellierung empfohlen. Eine Anwendung dieser Best Practice Patterns kann zu einem qualitativ hochwertigeren Diagramm führen.

Mbse4u.com ist eine weitere Webseite, die sich mit Model-Based Systems Engineering beschäftigt. Artikel auf dieser Webseite werden unteranderem von Tim Weilkiens geschrieben und beschreiben sprach Konstrukte der SysML und Empfehlungen für die Modellierung mit ihnen. Tim Weilkiens ist ein Berater, Trainer, Autor und Vorstandsmitglied von „oose“, zudem ist er ein aktiver Bestandteil der Object Management Group und dem International Council on Systems Engineering. Er kann mit einer Menge an Wissen bezüglich der SysML glänzen, da er Mitverfasser der SysML Spezifikation ist und zusammen mit Yves Bernard (Airbus) die Entwicklergruppe leitet, die die SysML v1 und die SysML v2 entwickeln (Weilkiens & Muggeo, The MBSE Podcast, 2023).

3 Methodik

Der Methodik Teil der Arbeit beschreibt auf welche Art und Weise und warum genau diese Forschung betrieben wurde. Zusätzlich wird erläutert, wie die Daten erhoben wurden, woher die Daten stammen und wie diese verarbeitet wurden. Im Anschluss gibt es eine Erläuterung zu allen, aus den verarbeiteten Daten, gebildeten Bewertungskriterien.

3.1 Art und Weise der Forschung

Um die Fragestellung dieser Arbeit zu untersuchen und zu beantworten, wurde ein Plugin für den Cameo Systems Modeler entwickelt, welches eine qualitative Nutzwert Analyse auf Strukturdiagramme anwendet. Eine qualitative Forschungsmethode wurde gewählt, um ein möglichst breites Abbild der etablierten Heuristiken für die Modellierung von Diagrammen in SysML zu erzeugen und möglichst viele Bewertungskriterien daraus abzuleiten.

Durch eine breitgefächerte Forschungsmethodik werden zusätzlich neue Zusammenhänge zwischen dem Analysieren von Diagramqualität und dem qualitativen Bewerten anderer Erzeugnisse erkenntlicher.

Ein anderes Erzeugnis ist zum Beispiel ein Java code Dokument aus dem Themenfeld der klassische Softwareentwicklung, welches auf sehr ähnlicher Weise bewertet werden kann. Weitere Themenfelder sind das Systems Engineering, das Model-Based Systems Engineering, das Objektorientierte Programmieren und die Agile Softwareentwicklung.

Um der Theorie nachzugehen, dass die Diagramqualität bewertbar ist und sich die Bewertung automatisieren lässt, wird außerdem deduktiv argumentiert.

3.2 Die Daten und ihre Analyse

Die erhobenen Daten stammen aus den verschiedensten Arten von Quellen. Analysiert wurden Daten aus Fachliteraturen, Büchern, verschiedenen online Enzyklopädien, Reporte, Studien, Zeitschriften und Websites.

Damit ein Datum als Bewertungskriterium eines Diagrammes in SysMl dienen kann, müssen sie einigen Anforderungen entsprechen. Zuerst müssen sie auf ein Diagramm oder dem Modellierungsprozess an sich anwendbar sein. Sie müssen zusätzlich das Diagramm Syntaktisch und oder semantisch aufwerten.

Natürlich müssen die Daten auch noch verarbeitet werden. Dafür wurden redundante Informationen ausgefiltert und im Falle von widersprüchlichen Heuristiken wird sich auf nur eine festgelegt. Alle anderen Informationen werden im Anschluss in ein Excel-Dokument gespeist (siehe Anhang 1: Excel Tabelle mit Bewertungskriterien).

Das Excel-Dokument ist unterteilt in folgenden Tabellen: Eine für Block Definition Diagramme, eine für interne Block Diagramme, eine für Paket Diagramme und eine Tabelle für generelle Heuristiken bezogen auf den Modellierungsprozess. Zusätzlich wird in den Tabellen der

Diagramme unterschieden, ob es sich um eine Heuristik handelt die Syntax oder Semantik verbessern soll.

Zu jeder Heuristik wird eine Erklärung abgegeben. Die Erklärung gibt Auskunft, wohin die Missachtung der Heuristik führen kann oder auch was sich durch Einhaltung verbessern wird. Bei vielen der Heuristiken bietet es sich zusätzlich an, eine weitere Kategorisierung vorzunehmen. Die hinzukommende Kategorisierung ist unterteilt in Drei Grade. Die Grade repräsentieren den Grad der Qualitätsminderung des Models bei Missachtung der jeweiligen Heuristik.

Die Drei Grade sind:

- Smell: Führt unter Umständen zu einem Qualitätsverlust der Modell Struktur und oder der Diagramme.
- Warning: Qualität der Diagramme und oder Modell Struktur ist bereits gemindert.
- Error: Semantik oder Syntax der Diagramme und oder der Modell Struktur ist fehlerhaft.

Die gewonnen Heuristiken dienen der Nutzwertanalyse als Bewertungskriterium. Die drei Grade der Qualitätsminderung repräsentieren die Entscheidungsmöglichkeiten, zusammen mit der Gewichtung des jeweiligen Bewertungskriteriums, bilden sie eine Bewertung, über die Qualität des Diagramms. Das lässt sich auch wunderbar als Plugin für den Cameo Systems Modeler implementieren und dadurch lässt sich die Nutzwertanalyse automatisiert auf Diagramme anwenden.

3.3 Die erhobenen Bewertungskriterien

Wie bereits erwähnt, wurde aus den gesammelten Heuristiken Bewertungskriterien abgeleitet. Im folgenden Teil werden die Bewertungskriterien genauer erläutert und auf die in Kapitel 2.2 erarbeiteten Konzepte zurückgeführt. Zusätzlich wird argumentiert, auf was für eine Art von Qualitätsminderung oder Verbesserung sich das Bewertungskriterium bezieht. Begonnen wird mit den Kriterien der BDD (Block Definition Diagram), dann die der IBD (Internal Block Diagram) und zum Schluss die der PKG (Package Diagram).

Eine Zusammenfassung aller erhobenen Kriterien befindet sich in Kapitel 5.1.

3.3.1 Bewertungskriterien für Block Definition Diagramme

Ein Bewertungskriterium für die Qualität eines Block Definition Diagrammes ist die Tiefe der „Composition-Associations“. Ähnlich wie bei der Vererbung in der objektorientierten Programmierung, führt eine zu Tiefe Komposition oder Aggregation zwischen Blöcken und ihren Einzelteilen zu einer Minderung der Übersichtlichkeit und der Flexibilität. Getreu nach dem Prinzip „Keep it simple stupid“ ist jenes Block Definition Diagramm zu bevorzugen, wessen Einzelteile des zu betrachtenden „System under Development“ auf der höchstmöglichen Kompositionshierarchie liegen.

Eine weitere Eigenschaft von Qualitativen BDD ist, dass Blöcke nur dann miteinander kommunizieren, wenn sie in direkter Beziehung zueinanderstehen. Der Grund dafür ist das

Gesetz von Demeter. Die Übersicht und die nach Vollziehbarkeit des Kontrollflusses und Objektflusses wird bewahrt, da die System Struktur die Kommunikation der Komponenten vorgibt. Es werden zusätzlich Schnittstellen eingespart und Redundanz unterbunden.

Eine von R. Martin erkannte Heuristik, die dem Buch Clean Code zu entnehmen ist, ist es die Anzahl von Argumenten einer Funktion so gering wie möglich zu halten. Die Ideale Anzahl von Argumenten ist 0 gefolgt von eins und zwei, mehr als 2 sollten vermieden werden. Das gilt für Input Argumente sowie Output Argumente (Martin R. C., Function Arguments, 2008). In der SysML sind die „Ports“ dafür da Information- und oder Datenfluss entgegenzunehmen oder zu senden, ganz ähnlich den Argumenten von Funktionen. Zu viele Input- und Output-Ports mindern die Übersicht des BDD und das Erstellen von Test-Cases wird zu einer immer schwereren Aufgabe. Dementsprechend ist die Qualität eines Diagrammes hochwertiger, wenn nur die nötigsten Informationen zwischen Blöcken ausgetauscht werden und die Ports auf ein Minimum reduziert sind.

Passend dazu, sollten Module ein so klein wie mögliches Interface anbieten (Martin R. C., Too much Information, 2008). Ein Modul kann in der SysML als das System under Development, betrachtet werden und das interface als die „Schnittstelle“ des Systems zu dem „Operationalen Kontext“. Dementsprechend sollte ein wohl definiertes System nur die nötigsten Schnittstellen für die Interaktion mit der Umgebung anbieten und dennoch viele Funktionalitäten mit wenigen Informationen ermöglichen. Damit lässt sich erreichen, dass sich Informationen besser verstecken lassen und sich so eine Verbesserung der Übersicht sichtbar macht. Zusätzlich sind einzelne Teile nicht so stark voneinander abhängig und das ermöglicht eine bessere Flexibilität und wieder Verwendbarkeit von Komponenten.

Auch das Einhalten von Namenskonvention tätig Aussage über die Qualität eines BDDs. Bei willkürlicher Benennung von Blöcken und ihren „Properties“ kann Verwechslungsgefahr entstehen und die Qualität wird gemindert. Aus diesem Grund hat sich Webel IT Australia überlegt, die Benennung von Blöcken mit einem „UpperCamelCase“ und Block Properties mit einem „lowerCase“ zu beginnen (Darren, Use 'UpperCamelCase' (a.k.a. PascalCase) names for Classifiers such as UML Classes and SysML Blocks) to avoid confusion with 'lowerCase' Property names [see however special naming conventions for acronyms and SysML contract and flow Blocks], 2023). Natürlich lässt sich diese Heuristik auch mit anderen Konventionen realisieren, doch um einer Vereinheitlichung der Modellierungsprinzipien näher zu kommen wird, in der Implementierung der Qualitätsanalyse als Plugin, dieser Ansatz gewählt.

Aus den gleichen Gründen empfiehlt Dr. Darren, dass man keine Property Namen nutzen soll, die identisch zu dem Namen des Besitzenden Blockes sind (Darren, DO NOT use Property names that are identical to the names of the Classifier (Class, DataType, Block, ValueType) that type them!, 2023).

Eine der größten Fehlerquellen, die zu einer Verletzung der System Syntax führen kann, ist das falsche Runterbrechen in System Komponenten oder anders gesagt das Missachten der Dekompositionsregeln. Die Regeln besagen, dass wenn eine neue Schnittstelle während der Dekomposition entsteht und diese an der Systemgrenze sichtbar ist, sie auch bei allen Eltern-Komponenten vorhanden sein muss. Ist sie nicht an der Systemgrenze sichtbar, kann sie als

„Feature Interaction“ genutzt werden und muss nicht in allen Eltern-Komponenten vorhanden sein. Mit diesen Regeln lassen sich Aussagen über die Qualität des Diagramms treffen, denn wurde die Dekomposition korrekt durchgeführt ist die Syntax des Diagramms objektiv wertvoller.

Zusätzlich hilft die Dekomposition die Elemente des BDD sinnvoll zu strukturieren, denn die Elemente des Diagrammes müssen methodisch positioniert werden, um sicherzustellen, dass das Diagramm gut strukturiert ist und Elemente effizient miteinander kommunizieren (Friedenthal, Moore, & Steiner, Diagram Layout, 2015). Dementsprechend bietet es sich an, die Anordnung der Elemente entsprechend der Dekompositionshierarchie vorzunehmen. Dazu muss man erwähnen, dass dies nicht der einzige Ansatz zur Positionierung von Elementen ist, aber in den meisten Fällen anwendbar und zu bevorzugen. Durch die gewonnene Struktur und Effizienz wird dem Verlust von Übersicht und der Gefahr vor Missverständnissen entgegengewirkt.

Ein weiteres Kriterium für die Qualität eines BDDs ist, dass jeder Block ein Internes Block Definition Diagramm besitzt. So ist eines der empfohlenen Best Practice Patterns der SysML Partner das rekursive Dekomponieren der Block Hierarchien mit abwechselnden BDD-Definitionen und IBD-Anwendungen (SysML, 2023). So trifft ein BDD mit Blöcken ohne zugehörigen IBDs nur Aussagen über die System Architektur, alias die Black Box Sicht auf das System, aber keine in Bezug auf die Semantik der Komponenten, ergo die Semantik ist nicht vorhanden.

3.3.2 Bewertungskriterien für Interne Block Diagramme

Da es sich bei BDDs und IBDs um sehr ähnliche Konzepte handelt, mit dem Unterschied, dass sie das System aus verschiedenen Viewpoints betrachten, nutzen sie teilweise identische Elementtypen. Aus diesem Grund lassen sich einige Designprinzipien der BDDs auf die IBDs übertragen.

Wie auch schon bei den BDDs sollten in den IBDs die Anzahl der Ports von den „Parts“ auf ein Minimum reduziert werden, das gilt für Input- sowie Output-Ports. Durch das Reduzieren auf nur notwendige Ports wird Redundanz entfernt, Verwechslungsgefahren und potenzielle Fehlerquellen vermieden, die Beziehungen zwischen Parts werden übersichtlicher und Komponenten gewinnen an wieder Verwendbarkeit. Auch hier gilt, ein IBD gewinnt an Qualität, sobald weniger Schnittstellen für die Kommunikation genutzt werden.

Um Informationsverlust vorzubeugen, sollten möglichst alle Parts des Blockes in dem IBD abgebildet sein. Bei komplexeren Projekten führt dies allerdings nicht immer zu einem Gewinn an Qualität. In so einem Fall kann es sich auch lohnen Parts zu verstecken, jedoch sollte man sich genau überlegen, ob die fehlende Information nicht doch relevant für den Nutzer des Diagramms sein könnte.

Auch bei kleineren Projekten kann das Verstecken von Informationen vorteilhaft sein. Sollte der Typ eines Ports oder Properties ausreichen, um die Funktionalität zu beschreiben, solltest

du überlegen, die Namen zu verstecken (Darren, UML/SysML: In Internal Block Diagrams: Consider hiding the name of a named Port or Property in a Diagram if its Type is sufficient to indicate its role., 2023). Im Kontrast dazu steht, dass wenn der Name eines Ports die einzigartige Funktionalität, und ein ItemFlow eines Connectors den Typ des Ports impliziert, solltest du überlegen den Typ auf dem Port Symbol zu verstecken (Darren, UML/SysML: In Internal Block Diagrams: If you have a Port with a name that indicates a unique role AND and if there is an ItemFlow on a Connector that implies or suggests the Type of the Port, consider hiding the Type on the Port symbol., 2023). Beide Empfehlungen haben das Ziel, durch das Entfernen von redundanten Informationen, Unordnung zu beseitigen. Für welchen Ansatz man sich entscheiden sollte, hängt von der Menge an redundanten Informationen ab, die entfernt werden. Sind also deskriptive Namen der Ports und Properties oder der Port Typ, um die Funktion oder Rolle zu beschreiben, redundant, ist die Qualität des Diagramms geschmälert.

Dr. Darren empfiehlt außerdem das Vermeiden von überlappenden „Connector Lines“ von einem Port ausgehend oder zu einem Port führend (Darren, Webel Best Practice: SysML/SysPhS: DO NOT use overlapping Connector lines from/to one Port (can be misunderstood as "summed" flow and/or physical node/fork/junction)., 2023). Eine Überlappung von Connector Lines kann zu Missverständnissen führen, da sie als ein „Summed-Flow“ und oder „Node“, „Fork“ und „Junction“ interpretiert werden kann. Aufgrund von einem erhöhten Risikofaktor bezüglich Missverständnisse ist eine Qualitätsminderung des Diagramms vorhanden. Dementsprechend sind IBDs qualitativ hochwertiger, wenn jeder Port ausschließlich eine Connector Line besitz, oder die Connector Lines durch das „Nesting“ von Ports aufgeteilt werden.

Aus den gleichen Gründen wie bei den BDDs, sollte die Positionierung der Parts eines IBDs, einer Methodik folgen. Die naheliegendste Lösung für eine gute Strukturierung ist das Positionieren der Parts entsprechend ihrer Zugehörigkeit. Ist also ein Part einem Part oder Block untergeordnet, sollte der entsprechende Part innerhalb des zugehörigen Blockes platziert werden. Durch das Verschachteln der Parts entsteht eine verbesserte Nachvollziehbarkeit der Blockstruktur und die Übersicht verbessert sich. Sollten sich Parts nicht innerhalb ihres zugehörigen Blockes befinden, ist die Qualität des Diagramms vermindert, da ohne zusätzliche abgebildete Beziehungen nicht erkenntlich wird, welchem Block sie angehören.

Ports in einem IBD bilden das syntaktische Interface für einen Block und verbinden miteinander interagierende Parts. Ein Port ohne Semantik trifft keine Aussage bezüglich der Interaktion zwischen Parts und ist deswegen mit einem Verlust an Informationen verbunden. Zusätzlich sollte darauf geachtet werden, dass nur eine Art von Port innerhalb des Diagramms genutzt wird, in den meisten Fällen wird dazu empfohlen ausschließlich Proxy-Ports zu nutzen. So zum Beispiel weist Weilkiens darauf hin, nur Proxy-Ports zu nutzen und Full-Ports zu ignorieren, da das Nutzen nur einer Art von Ports die Stereotype Notation der Ports überflüssig macht und dadurch das Diagramm weniger überladen wirkt (Weilkiens, SysML Full Ports versus Proxy Ports, 2023).

Ein Proxy-Port muss zudem immer, durch einen Interface Block typisiert werden, wird dies missachtet, wird die Syntax der SysML verletzt (Object Management Group, 2019). Ein weiteres Kriterium für semantisch wertvolle Ports ist, dass der Interface Block eine „Flow Property“ besitzt. Ohne Flow Property ist nicht klar, ob es sich um eine Input Schnittstelle oder Output Schnittstelle handelt, dadurch entsteht eine Verwechslungsgefahr, und ein Verlust an Semantik entsteht.

Eine weitere Empfehlung von Dr. Darren ist es Parts, die miteinander kommunizieren, über eine Assoziation miteinander sichtbar zu machen (Darren, If instances of Blocks “communicate” with each other use an Association between them., 2023). Daraus lässt sich ableiten, dass Ports zumindest einer Connector Line zugeordnet sein sollten. Sind Ports ohne Konnektoren vorhanden ist die Qualität der Struktur des Diagrammes, aufgrund von Informationsverlusten, geschmälert.

3.3.3 Bewertungskriterien für Packet Diagramme

Die Bewertungskriterien für Paket Diagramme unterscheiden sich von Kriterien der BDD und IBD in dem Aspekt, dass die Sicht nicht das System oder die Komponenten beschreibt, sondern die Modellstruktur an sich. Deswegen werden in Paket Diagrammen nicht nur die Elemente eines Diagrammes und das Diagramm an sich betrachtet, sondern auch die Strukturierung der Diagramme untereinander, oder anders gesagt, Empfehlungen für die Modellierung von Paket Diagrammen wirken sich nicht nur auf die Qualität des Diagramms aus, sondern auch auf die Qualität der gesamten Modellstruktur.

Zudem lässt sich die Einhaltung von Empfehlungen an die Modellstruktur nur schwer automatisiert überprüfen und als ein Maß der Qualitätsbewertung nehmen, da jedes Projekt und/ oder Unternehmen eine andere Herangehensweise, an das Strukturieren der Modellhierarchie, verfolgt.

Eine starke und die einzige Empfehlung, dieser Forschungsarbeit, bezüglich der Modellstrukturierung ist es, ein „Top-Level“ Paket Diagramm in der Modellstruktur anzulegen. Eine der Webel Best-Practices ist es nämlich, die meisten Elemente nicht unter der Projektwurzel anzulegen (Darren, MagicDraw/Cameo: Keep most elements out of the top-level of a project (i.e. not directly under the project Root). Have one top-level index/content/package diagram., 2023). So ein Top-Level Diagramm verbessert die Struktur des Gesamten Projektes und verbessert damit die Übersicht.

Sollte es sich nicht anbieten ein Top-Level Diagramm zu erstellen, so sollte zumindest ein Paket Diagramm existieren, welches das Modell mittels weiteren Unterpaketen strukturiert, dies wird normalerweise mit dem Top-Level Diagramm gemacht. Das Strukturieren des Modells über die Pakethierarchie wird wichtiger, je größer die Anzahl der Elemente wird. Es ist offensichtlich, dass komplexe Modelle schnell, schwer zu handhaben sind, wenn die Einteilung in Unterpaketen nicht stattfindet. Eine gute Modellhierarchie erlaubt den Modellierern ein einfacheres Wiederverwenden von Elementen, eine bessere Navigation innerhalb des Projektes, verbesserter Datenaustausch zwischen Strukturen, und andere

Schlüsselaspekte des Entwicklungsprozesses (Friedenthal, Moore, & Steiner, Organizing a Package Hierarchy, 2015).

Wie auch schon bei den anderen beiden Strukturdiagrammen, sollte die Modellhierarchie nach einem Satz von Organisationsprinzipien modelliert werden. Zwei mögliche Ansätze werden in dem Buch A Pracitcal Guide to SysML vorgestellt. So sollte die Modellhierarchie nach der Systemhierarchie oder nach den verschiedenen Phasen des Produktlebenszyklus erfolgen (Friedenthal, Moore, & Steiner, Organizing a Package Hierarchy, 2015).

Ein Merkmal, welches über die Qualität eines Paket Diagramms entscheidet, ist das Nutzen von „Containments“ innerhalb des Diagrammes. Ist die Modellstruktur entschieden sollte diese in einem Diagramm festgehalten werden. Besitzt ein Paketdiagramm keine Containments, wird der Anschein erweckt, dass sich jedes Element des ganzen Modells auf nur einer Hierarchieebene befindet. Dies wirkt sich negativ auf den Informationsgehalt und der Übersicht aus, zudem entsteht eine Verwechslungsgefahr, sollten sich zwei Pakete mit identischen Namen aber in unterschiedlichen Namespaces befinden. Containments können auf zwei Arten dargestellt werden. Typischerweise werden Containments über einem Fadenkreuzsymbol, von dem mehrere Pfade ausgehen, als eine Art Baumstruktur dargestellt oder über das Positionieren von Modellelementen innerhalb des Paketsymbols (Friedenthal, Moore, & Steiner, Organizing a Package Hierarchy, 2015).

Da Pakete auch als Namespaces für die beinhalteten Elemente dienen, sollten Packages, nach dem was die enthaltenden Elemente darstellen sollen, benannt werden. Das Benennen nach Inhalt oder Funktionalität kann allerdings nur sehr schwer programmatisch erfasst werden und dadurch in den meisten Fällen nicht als ein automatisiertes Bewertungskriterium implementiert werden. Falls allerdings zwei Pakete auf dem gleichen Projekt-Level identisch benannt wurden, kann das zu einer Verletzung der Einzigartigkeitsregel der Element Namen innerhalb seines Namespaces führen (Friedenthal, Moore, & Steiner, Packages as Namespaces, 2015) und so als ein maß der Diagramm Qualität genutzt werden. Sind also zwei Pakete mit identischem Namen auf der gleichen Projektebene, führt dies zu einer Verletzung der Syntax und dadurch zu einer Qualitätsminderung der Projektstruktur.

3.4 Gütekriterien

Ich versichere zusätzlich, dass die Gütekriterien qualitativer Forschung eingehalten wurden. Durch das detaillierte Erläutern des Prozesses dieser Forschung und der resultierenden nach Vollziehbarkeit ist das Gütekriterium der Transparenz eingehalten. Damit diese Forschung auch das Kriterium der Intersubjektivität erfüllt, wurden die Ergebnisse angemessen reflektiert. Zusätzlich gibt es verschiedene Interpretationsmöglichkeiten der Ergebnisse, die auch vorgestellt wurden. Das dritte Gütekriterium die Reichweite, ist auch erfüllt, da bei gleichem Forschungsvorgang die gleichen Ergebnisse erzielt werden würden.

4 Ein Plugin für die automatisierte Einstufung des Verletzungsgrades von Bewertungskriterien

Dieses Kapitel beschäftigt sich mit der Implementierung der Qualitätsevaluation von einigen Bewertungskriterien mit der Cameo Systems Modeler API (Application Programming Interfaces) und der Beispielhaften Anwendung auf Diagramme.

Die Rückmeldung des Plugins an den Nutzer, liefert eine Einstufung des Verletzungsgrades der implementierten Überprüfungen. Die Einstufung, kann als Bewertungsmaßstab genutzt werden, und durch Anwendung auf die Bewertungskriterien und ihrer Gewichtung, den Nutzwert des Diagrammes bestimmen.

Um den vorgestellten Code zu verdeutlichen, wird Schritt für Schritt ein Diagramm modelliert indem bewusst Fehler eingebaut werden. In jedem Schritt wird der zu veranschaulichende Code genutzt, um den Modellierer anzuzeigen, was für eine Qualitätsminderung vorhanden ist und in welchem Ausmaß.

Eine Zusammenfassung aller implementierten Funktionen befindet sich in Kapitel 5.1.

4.1 Modellierung eines minderwertigen BDD's und die Bewertung des Plugins

Der Folgende Code Ausschnitt ist eine Funktion, die überprüft, ob ein gegebenes Element mit einem Lower Case beginnt. Ist dies der Fall, wird ein True zurückgegeben.

```
1.        // Returns True, if the given PresentationElement Starts with a Lower Case
2.        // Else Returns False
3.        public boolean checkIfPresentationElementStartsWithLowerCase(PresentationElement
presentationElement){
4.                  char first = presentationElement.getName().charAt(0);
5.
6.                  if(first == Character.toLowerCase(first) && first !=
Character.toUpperCase(first))
7.                            return true;
8.
9.        return false;
10.       }
```

Die Abbildung 1 zeigt das Nutzen des Plugins auf den Block „suD_WorstCase". Da der mit dem Plugin selektierten Block mit einem kleinen Buchstaben beginnt, wird der Nutzer über ein Smell informiert.

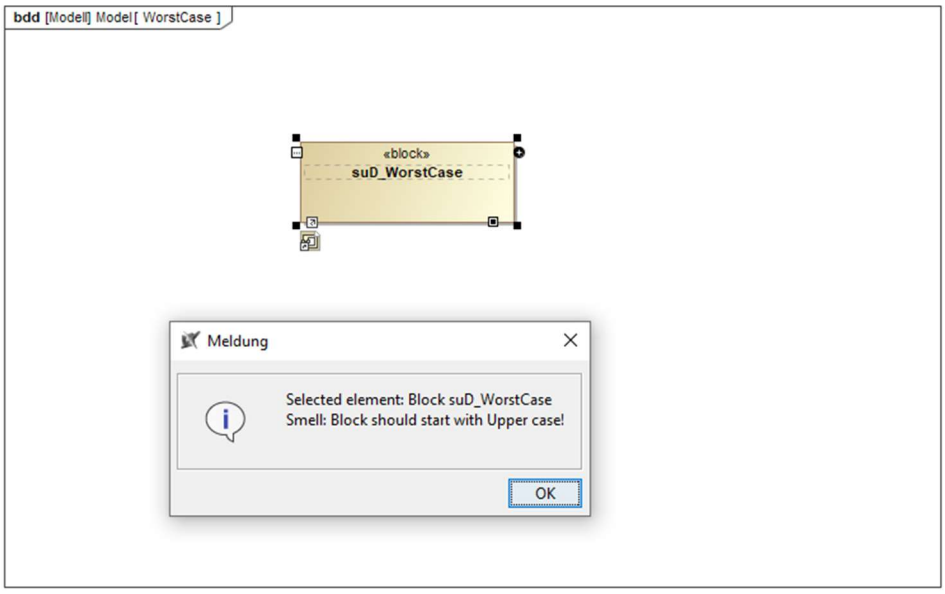

Abbildung 1: Blöcke starten mit einem Lower Case

Die Funktion „checkIfAllProxyPortsHaveAType“ tut genau dies. Als Argumente nimmt sie eine Liste von ProxyPorts und überprüft innerhalb einer Schleife alle Ports auf ihre Typisierung. Ist ein Port vorhanden, der keinen Type besitzt, wird ein false zurückgegeben.

```
1.      // Checks if all Ports of a given List of ProxyPorts do have a specified Type
2.      // Returns True if all Ports have a specified Type
3.      // Returns False if one Port does not have a specified Type
4.      public boolean checkIfAllProxyPortsHaveAType(List<PresentationElement> listProxyPorts)
{
5.              for(int i = 0; i < listProxyPorts.size(); i++) {
6.                      var port = (Port) listProxyPorts.get(i).getElement();
7.
8.                      if(port.getType() == null)
9.                              return false;
10.             }
11.             return true;
12.     }
```

In Abbildung 2 wird die Anwendung der Funktion, für die Überprüfung der Typisierung, verdeutlicht. Der selektierte Block ist das „suD_WorstCase“ mit den Proxy Port „port1“. Das Plugin bedient sich aller Ports des Blocks und gibt diese als Liste an die Funktion weiter. Da der Port keinen Type besitzt, wird in der Ausgabe mithilfe des Boolean Werts den Nutzer eine Warning angezeigt.

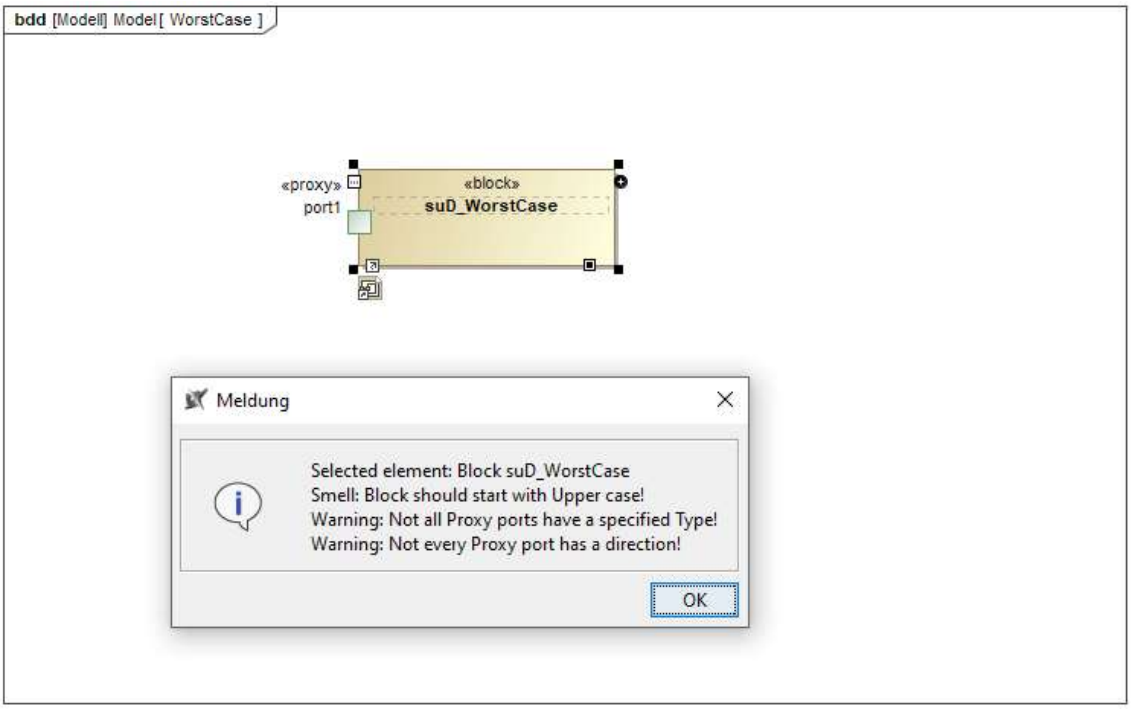

Abbildung 2: Nicht jeder Port eines Blockes ist typisiert

Das nächste Code Beispiel, dient der Überprüfung, welche Art von Ports genutzt wurden. Zuerst werden alle Proxy Ports, Full Ports und Ports eines selektierten Elements als Listen gesammelt. Die Anzahl der Elemente innerhalb der Listen wird bestimmt, um in der Ausgabe angeben zu können ob verschiedene Arten von Ports und oder ob nur Proxy Ports genutzt wurden.

```
1. // Get All Ports from selected Block
2. var proxyPorts = getAllProxyPortsFromPresentationElement(presentationElement);
3. var fullPorts = getAllFullPortsFromPresentationElement(presentationElement);
4. var ports = getAllPortsFromPresentationElement(presentationElement);
5.
6. if(fullPorts.size() > 0 || ports.size() > 0)
7.     buffer.append("Smell: Every Port in a BDD should be a Proxy port! \n");
8. if(proxyPorts.size() > 0 && fullPorts.size() > 0 || proxyPorts.size() > 0 && ports.size() > 0
|| fullPorts.size() > 0 && ports.size() > 0)
9.     buffer.append("Error: Different types of Ports have been used! \n");
```

Das Diagramm wurde in Abbildung 3 nun um einen Full Port erweitert. Mithilfe des vorherigen Code Beispiels, wird dem Nutzer ein Error und ein weiterer Smell angezeigt. Der Error entsteht durch das Benutzen verschiedener Arten von Ports in einem Diagramm und der Smell, weil es sich bei den Ports nicht ausschließlich um Proxy Ports handelt.

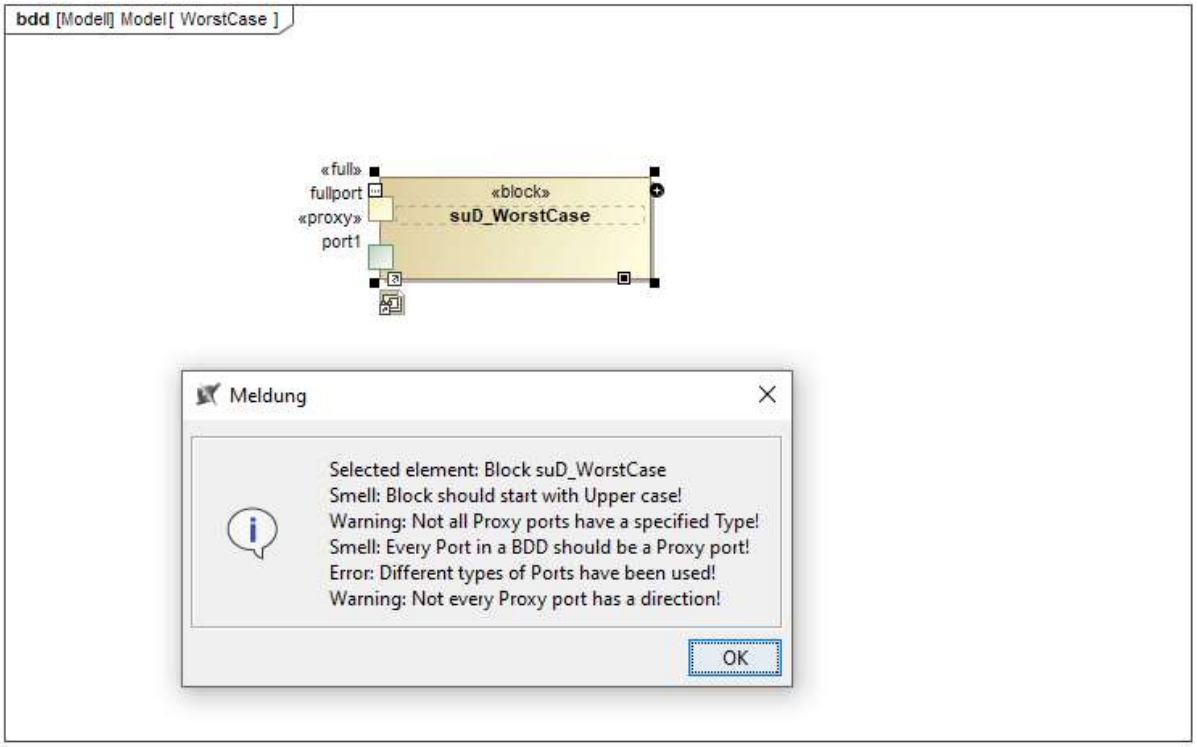

Abbildung 3: Nicht jeder Port ist ein ProxyPort, verschiedene Typen von Ports wurden benutzt.

In Abbildung 2 ist eine weitere Warning aufgetreten, auf die bisher noch nicht eingegangen wurde. Dafür ist die Folgende Funktion verantwortlich. Die Funktion „countProxyInputAndOutput“ nimmt eine Liste von Proxy Ports entgegen und zählt die Anzahl an Input Ports, Output Ports und Ports ohne eine Richtung. In Abbildung 2 wurde der nicht typisierte Port „port1“ hinzugefügt. Ohne eine Typisierung kann kein Flow Property vorhanden sein und dementsprechend auch keine Richtung.

```
1.      // Counts the Different Directions of a given List of proxyPorts
2.      // Returns an Array
3.      public int[] countProxyInputAndOutput(List<PresentationElement> proxyPorts){
4.          // count[0] : no Direction
5.          // count[1] : Input Port
6.          // count[2] : Output Port
7.          int count[] = new int[3];
8.
9.          // loop through all Proxy Ports
10.         for(int i = 0; i < proxyPorts.size(); i++) {
11.             var port = (Port) proxyPorts.get(i).getElement();
12.
13.             // Proxy Port does not own an Interface
14.             if(port.getType() == null)
15.                 count[0]++;
16.
17.             else {
18.                 // get the used Interface of the Proxy Port
19.                 var interfaceBlock = (Class) port.getType();
20.
21.                 // get attributes of the Interface Block
22.                 var interfaceBlockOwnedAttribute = interfaceBlock.getOwnedAttribute();
23.
24.                 // loop through all Attributes from the Interface Block
25.                 for (Property prop : interfaceBlockOwnedAttribute) {
26.
27.                     // get TaggedValues
28.                     var taggedValues = prop.getTaggedValue();
29.
30.                     // loop through all TaggedValues
31.                     for (var taggedValue : taggedValues) {
32.                         var value = taggedValue.getValue();
33.
34.                         // loop though all Values of the taggedValues
35.                         for (var val : value) {
36.                             // Flow Property is in
37.                             if (((EnumerationLiteral) val).getName().equals("in"))
38.                                 count[1]++;
39.                             // Flow Property is out
40.                             if (((EnumerationLiteral) val).getName().equals("out"))
41.                                 count[2]++;
42.                         }
43.                     }
44.                 }
45.             }
46.         }
47.         return count;
48.     }
```

Die Funktion liefert ein Array „count“ zurück, welches die Anzahlen der Ausgangs Ports, Eingangs Ports, und Ports ohne eine Richtung beinhaltet. Diese werden genutzt, um in der Ausgabe den Nutzer zu informieren, ob eine Qualitätsminderung vorhanden ist.

```
1.       if(counterOfDifferentDirectionalProxyPorts[0] > 0)
2.             buffer.append("Warning: Not every Proxy port has a direction! \n");
3.       if(counterOfDifferentDirectionalProxyPorts[1] >= 5)
4.             buffer.append("Error: Number of input Ports exceeded 4! \n");
5.       else if(counterOfDifferentDirectionalProxyPorts[1] >= 4)
6.             buffer.append("Warning: Number of Input Ports exceeded 3! \n");
7.       else if(counterOfDifferentDirectionalProxyPorts[1] >= 3)
8.             buffer.append("Smell: Number of Input Ports exceeded 2! \n");
9.       if(counterOfDifferentDirectionalProxyPorts[2] >= 4)
10.            buffer.append("Error: Number of output Ports exceeded 3! \n");
11.      else if(counterOfDifferentDirectionalProxyPorts[2] >= 3)
12.            buffer.append("Warning: Number of output Ports exceeded 2! \n");
13.      else if(counterOfDifferentDirectionalProxyPorts[2] >= 2)
14.            buffer.append("Smell: Number of output Ports exceeded 1! \n");
```

Abbildung 4 verdeutlicht die Verwendung der Funktion „countProxyInputAndOutput“, durch das hinzufügen mehrerer Eingangs Ports und Ausgangs Ports. Das Plugin zeigt zwei Smells an, da die Anzahl Eingangs Ports Zwei übersteigt und die Anzahl Ausgangs Ports höher als Eins ist. Würde sich die Anzahl an Ports weiter erhöhen, so würde aus einem Smell ein Warning und dann ein Error werden.

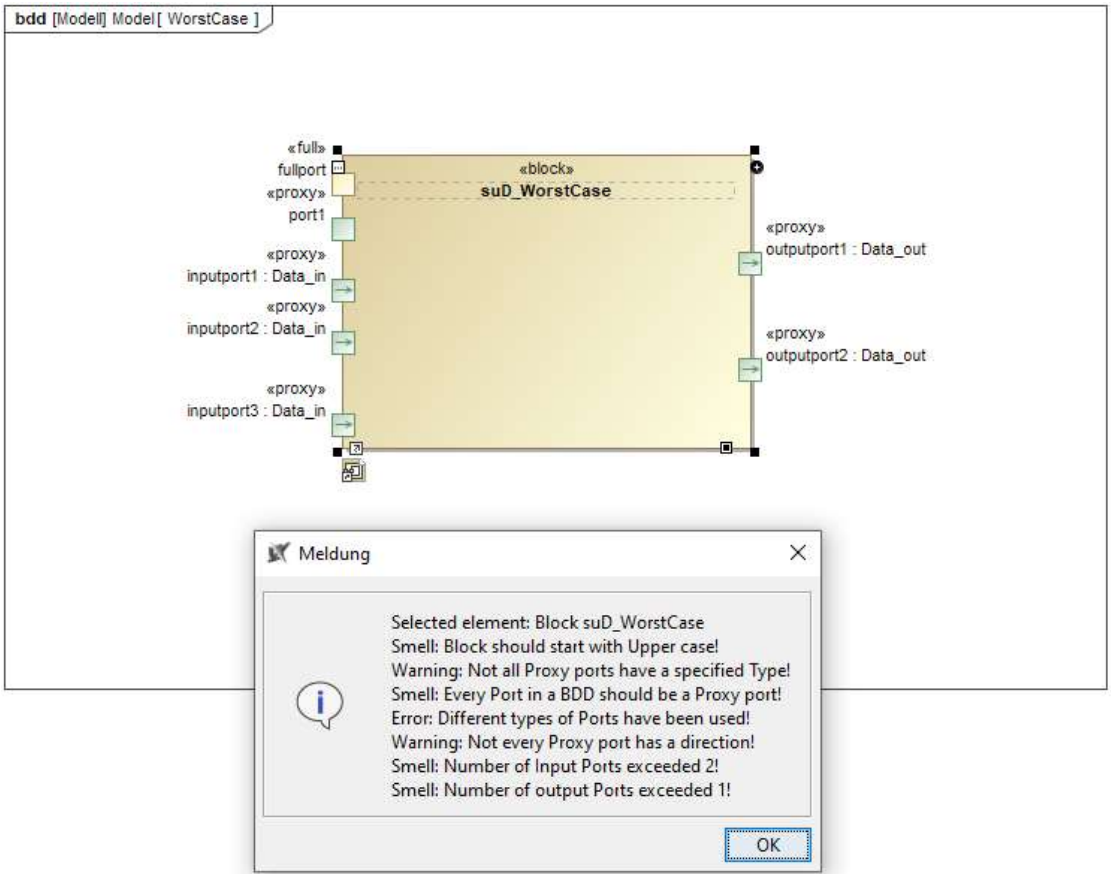

Abbildung 4: Zu viele Input-, Output-Ports. Nicht jeder Port besitzt eine Richtung

Eine weitere Funktionalität, die das Plugin bereitstellt, ist das Überprüfen auf eine korrekte Dekomposition. Die vorgestellte Funktion „checkIfDecompositionIsDoneRightInAllAboveBlocks“ nimmt einen Block entgegen (vorzugsweise der Block in der untersten Hierarchieebene) und überprüft mithilfe der wiederholten Aufrufung der Funktion „checkIfDecompositionIsDoneRightInBlockAbove“, ob die Interface Ports des System Under Developments, die auch in dem entgegengenommenen Block vorkommen, in allen dazwischen liegenden Komponenten vorhanden sind. Wird dies wiederholt auf jede Komponente des Systems Under Development angewendet und kein einziges False wird zurückgegeben, ist die Dekomposition korrekt.

```
 1.     // Check if the Decomposition of a given Block is done Right in all above Blocks
 2.     public boolean checkIfDecompositionIsDoneRightInAllAboveBlocks(PresentationElement
presentationElement){
 3.         var systemUnderDevelopment = getSystemUnderDevelopment(presentationElement);
 4.
 5.         // As long as the block to Check is not the System Under Development
 6.         while(!presentationElement.equals(systemUnderDevelopment)){
 7.             var proxyPorts = getAllProxyPortsFromPresentationElement(presentationElement);
 8.             if(!checkIfDecompositionIsDoneRightInBlockAbove(proxyPorts))
 9.                 return false;
10.             presentationElement = getBlockAbove(presentationElement);
11.         }
12.         return true;
13.     }
14.
15.       // Checks if the Functional Interfaces from a given List of ProxyPorts from a Selected
Block are existent in Parent Block
16.     public boolean checkIfDecompositionIsDoneRightInBlockAbove(List<PresentationElement>
proxyPorts) {
17.         var functionalInterfaces = getAllFunctionalInterfaces(proxyPorts);
18.
19.         // No Functional Interfaces have been found return true
20.         if(functionalInterfaces.size() == 0)
21.             return true;
22.
23.         var block = functionalInterfaces.get(0).getParent();
24.         var blockAbove = getBlockAbove(block);
25.
26.         // Block is already the System under Development return true
27.         if(blockAbove == block)
28.             return true;
29.
30.         // Iterate through all the functional Interfaces
31.         for(var functionalInterface : functionalInterfaces) {
32.             boolean functionalInterfaceHasBeenFound = false;
33.             var proxyPortsBlockAbove = getAllProxyPortsFromPresentationElement(blockAbove);
34.
35.             // Block Above does not have any Proxy Ports return false
36.             if(proxyPortsBlockAbove.size() == 0)
37.                 return false;
38.
39.             // Iterate through all proxyPorts from the Block above and check if one of the
proxyPorts match the functional Interface
40.             for(var proxyPortBlockAbove : proxyPortsBlockAbove) {
41.
if(proxyPortBlockAbove.getHumanName().equals(functionalInterface.getHumanName()))
42.                     functionalInterfaceHasBeenFound = true;
43.             }
44.
45.             // Functional Interface has not been found in the block above return false
46.             if(!functionalInterfaceHasBeenFound)
47.                 return false;
48.         }
49.
```

```
50.         // All Checks were successful
51.         return true;
52.     }
```

Das System Under Development wurde nun um 2 Komponenten erweitert, die jeweils mittels einer Gerichteten Beziehung zueinanderstehen und Unterkomponenten repräsentieren. Der „Component_1_1“ wurde zusätzlich der Proxy Port „inputport2“ hinzugefügt. Dieser Proxy Port ist auch in dem SuD vorhanden und deswegen ein Funktionales Interface. Eine Überprüfung des Blockes Component_1_1 mit dem Plugin liefert einen Error. Der Error ist aufgrund Fehlens des Interface Ports inputport2 in der „Component_1“ entstanden und dies eine Verletzung der Dekompositionsregeln ist.

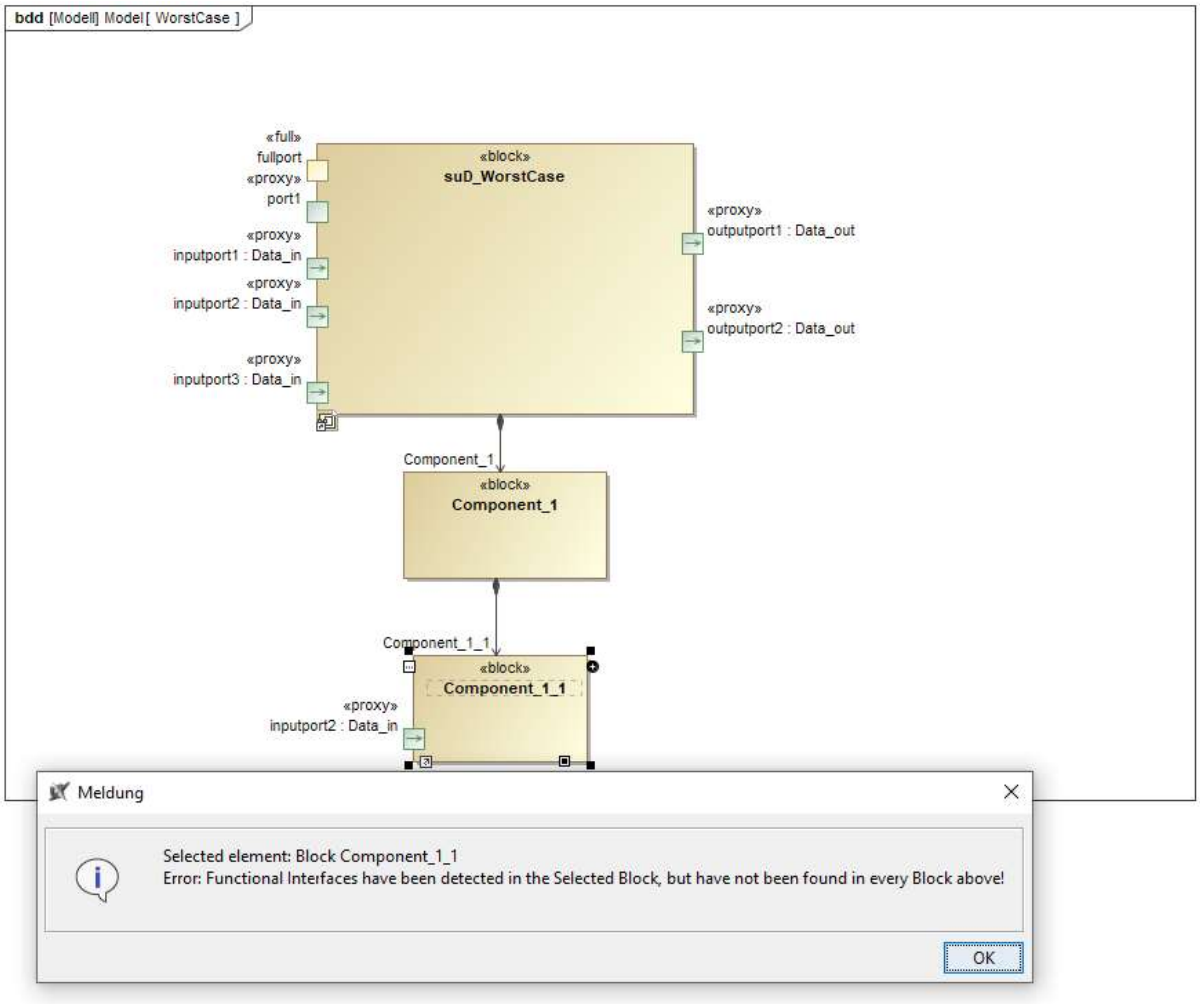

Abbildung 5: Dekomposition ist nicht korrekt

Die nächste Funktionalität des Plugins ist, dass erfassen der Dekompositionstiefe. Die Funktion „getNestingDepthFirstSearch“ nimmt einen Block entgegen und berechnet von dort ausgehend, über die Gerichteten Beziehungen hinweg, den Pfad zum Tiefsten Element. Dafür bedient sich die Funktion einer Rekursiven Tiefensuche.

```
1.      // Recursive tree traversal in DFS to find the deepest connected PresentationElement,
starting from the given PresentationElement
2.      // Returns level of the deepest PresentationElement
3.      public int getNestingDepthFirstSearch(PresentationElement presentationElement){
4.          var pathElementsGoingDown = getPathsGoingDown(presentationElement);
5.
6.          // Best case
7.          if(pathElementsGoingDown.size() == 0)
8.              return 0;
9.
10.         int levelOfDeepestPresentationElement = 0;
11.
12.         for(int i = 0; i < pathElementsGoingDown.size(); i++) {
13.             var child = pathElementsGoingDown.get(i).getSupplier();
14.             levelOfDeepestPresentationElement = Math.max(levelOfDeepestPresentationElement,
getNestingDepthFirstSearch(child));
15.
16.         }
17.         return levelOfDeepestPresentationElement + 1;
18.     }
```

Das berechnete Level, der Tiefsten Komponente, wird schließlich von der Funktion an die Ausgabe weitergegeben und für den Nutzer verarbeitet.

```
1.          if(levelOfDeepestPresentationElement >= 7)
2.              buffer.append("Error: Depth of composed associations exceeded 6! \n");
3.          else if(levelOfDeepestPresentationElement >= 5)
4.              buffer.append("Warning: Depth of composed associations exceeded 4! \n");
5.          else if(levelOfDeepestPresentationElement >= 3)
6.              buffer.append("Smell: Depth of composed associations exceeded 2! \n");
```

Dem Fehlerbehafteten Diagramm wird nun eine weitere Komponente hinzugefügt, welche die Tiefe der Gerichteten Beziehungen erhöht. Das Plugin zeigt nun einen Smell an, da die Tiefe einen Wert von Zwei überstiegen hat.

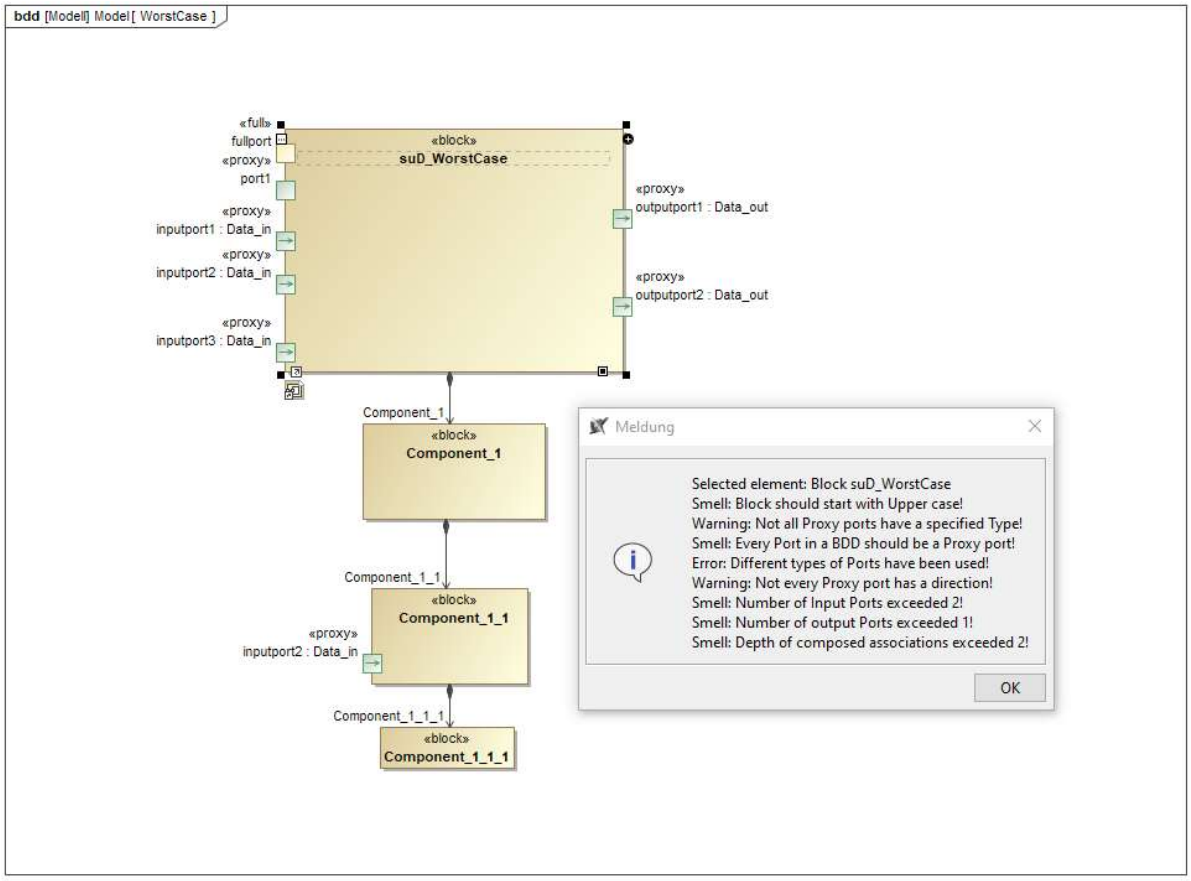

Abbildung 6: Die Tiefe der Kompositions Assoziation fängt an zu groß zu werden.

Die letzte Funktionalität des Plugins, bezüglich der Analyse von BDDs ist das Überprüfen des System Under Development auf ein zugehöriges IBD. Die Funktion “checkIfSuDHasDiagram“ nimmt einen Block entgegen und holt sich dessen höchste Elternkomponente (im Normalfall das SuD). Dann wird geprüft, ob dieses ein Diagramm vom Typen IBD besitzt. Besitzt es mindestens eins, so wird ein True ausgegeben.

```
1.     // Checks if the System Under Development (The Highest Block in the hierarchy) owns a
Diagram
2.     // Returns True if it Owns a Diagram
3.     // Returns False if it does not Own a Diagram
4.     public boolean checkIfSuDHasADiagram(PresentationElement presentationElement) {
5.         var systemUnderDevelopment = getSystemUnderDevelopment(presentationElement);
6.         var systemUnderDevelopmentElement = systemUnderDevelopment.getElement();
7.
8.         var ownedDiagrams = getOwnedDiagramsOfElement(systemUnderDevelopmentElement);
9.
10.        if(ownedDiagrams.size() == 0)
11.            return false;
12.
13.        return true;
14.    }
```

In der Abbildung 7 wurde der Block suD_WorstCase mit dem Plugin ausgewählt. Die Meldung zeigt nun einen weiteren Smell an nämlich, dass das zugehörige System Under Development dieses Blocks kein IBD besitzt. Die gleiche Meldung würde auch erzeugt werden, wenn das Plugin auf eine der Unterkomponenten angewendet wurde.

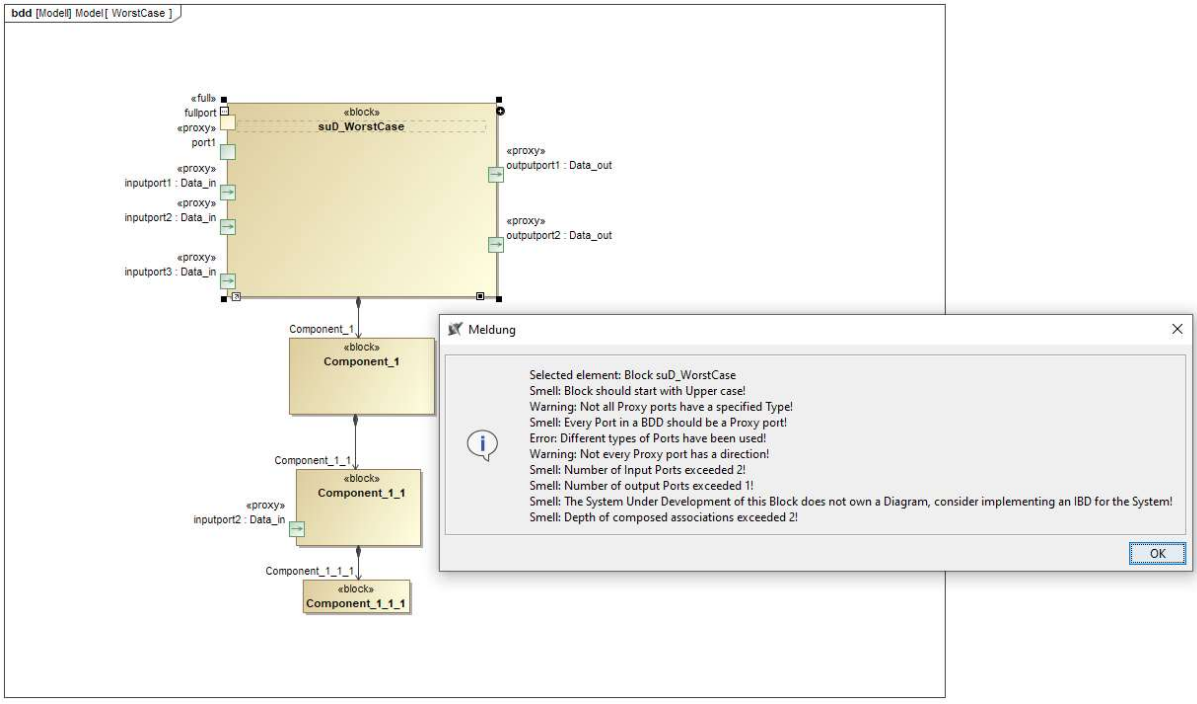

Abbildung 7: Das SuD besitzt kein IBD

4.2 Modellierung eines minderwertigen IBDs und die Bewertung des Plugins

Das Plugin bietet, zusätzlich zu den Überprüfungen der BDDs, Funktionen für die Bewertung von IBDs. Diese werden anhand des bereits erstellten Worstcase Szenarios vorgestellt. Abbildung 8 zeigt dementsprechend das IBD des suD_WorstCase Blocks.

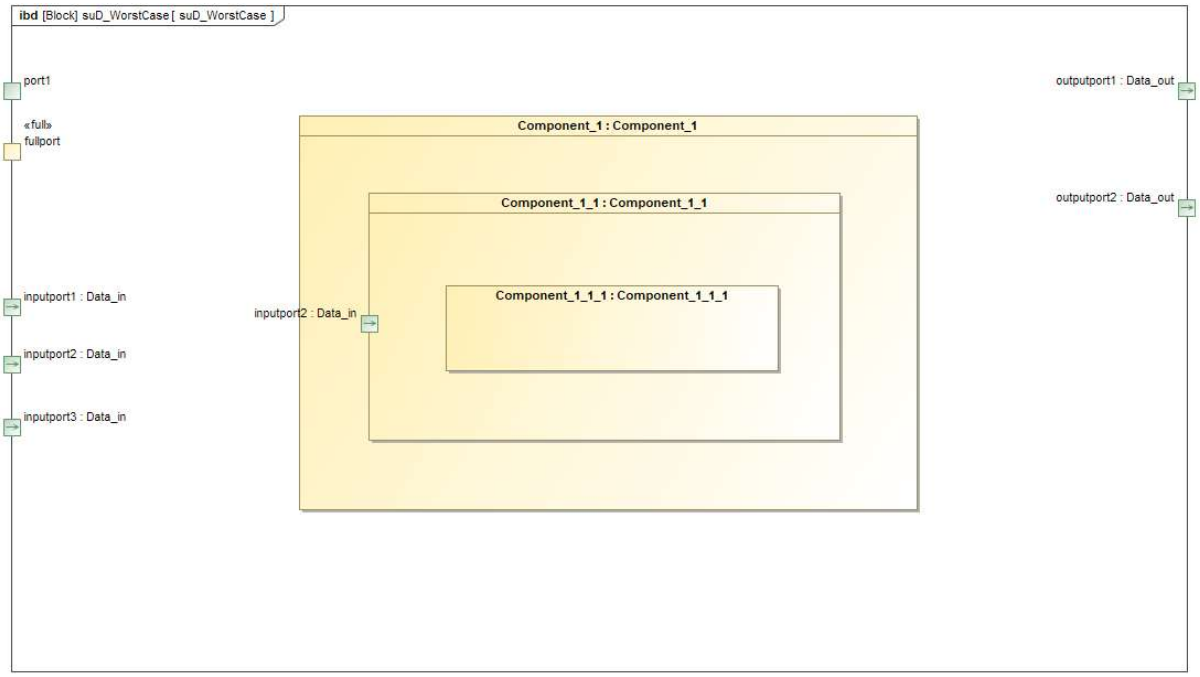

Abbildung 8: IBD des WorstCase SuD

Da sich beide Diagrammtypen Blockelemente zu Nutze machen, um Komponenten und Parts zu modellieren, lassen sich einige Funktionen für die BDDs auch auf die IBDs anwenden. So wird in Abbildung 9 das Part Property „Component_1“ mit dem Plugin selektiert, woraufhin ein Smell in der Meldung angezeigt wird. Der Smell entsteht aufgrund der bereits vorgestellten „checkIfPresentationElementStartsWithLowerCase“ Funktion. Bei Anwendung der Funktion auf Blöcke in IBDs, ist die Logik im Gegensatz zu der Anwendung in BDDs vertauscht. Denn Parts sollten mit einem kleinen Buchstaben beginnen.

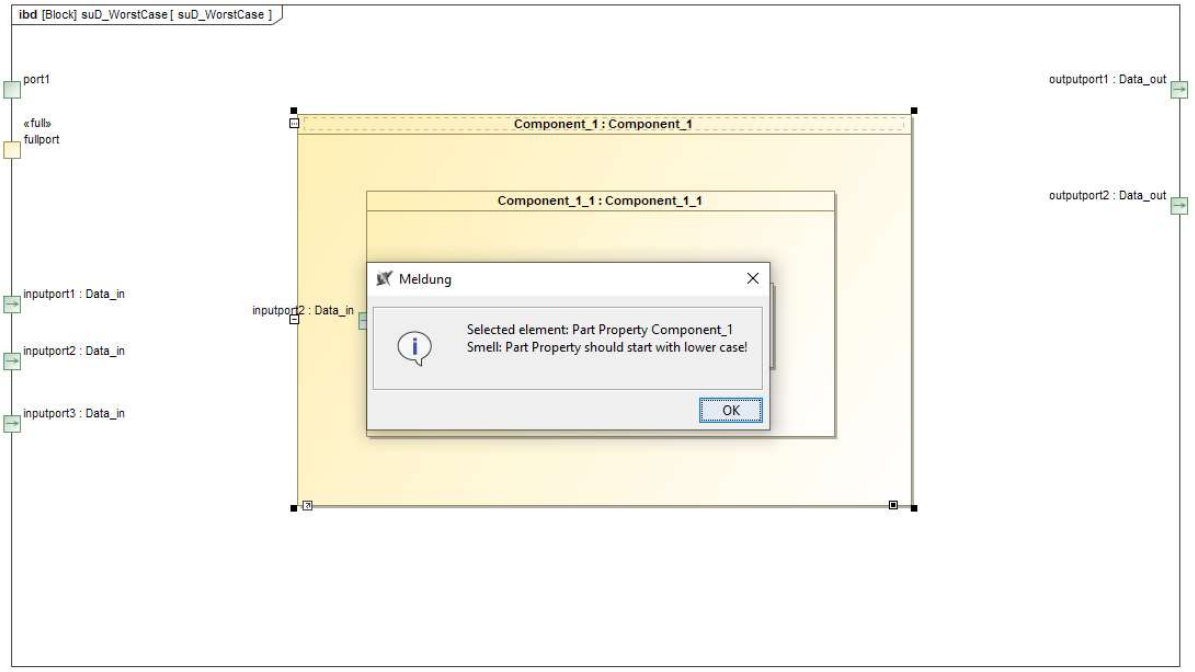

Abbildung 9: Part Property im IBD startet mit einem UpperCamelCase

Auch in IBDs ist die Typisierung der Ports wichtig. In Abbildung 10 wurde der Component_1 ein weiterer Port „inputport2“ hinzugefügt. Das Plugin wurde daraufhin auf der Component_1 angewendet. In dem Beispiel informiert das Plugin den Nutzer über Drei neue Warnings. Zwei Warnings sind aufgrund von bereits vorgestellten Funktionen entstanden. So liefert die Funktion checkIfAllProxyPortsHaveAType ein False an die Ausgabe, da der neue Port nicht typisiert ist. Die Funktion countProxyInputAndOutput liefert ein Array, in dem die Menge von Input Ports und Output Ports des Blocks gespeichert ist, da der inputport2 allerdings kein Flow Property besitzt, wird er als ein Port ohne Richtungsangabe gezählt und entsprechend eine Warning ausgegeben.

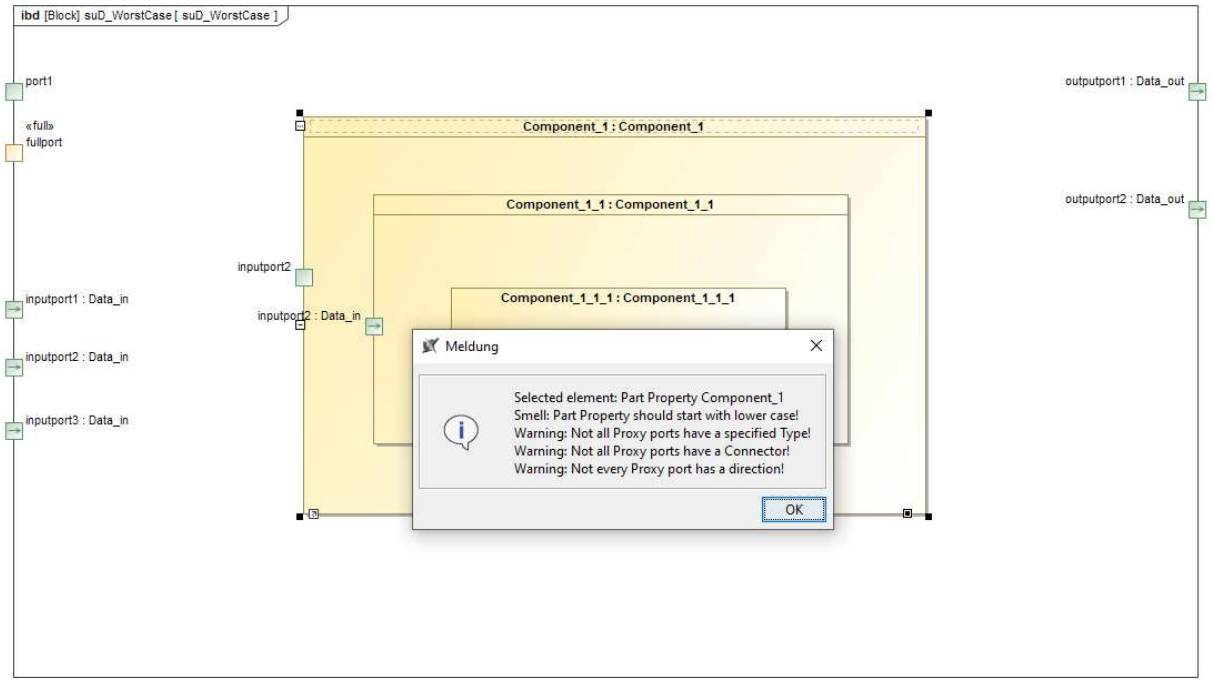

Abbildung 10: Nicht jeder Port im IBD ist typisiert und besitzt ein Flow Property. Zusätzlich fehlt eine Connector Line

Die dritte Warning ist durch eine neue Funktion entstanden und lässt sich nur auf Blöcke von IBDs anwenden, da die Connector Lines nicht von Relevanz für BDDs sind. Die Funktion „checkIfAllProxyPortsAreConnected“ nimmt eine Liste von Proxy Ports entgegen, und überprüft alle Ports, ob sie ein Pfadelement besitzen. Ist dies nicht der Fall besitzt der Port keine Connector Line. Ein Port ohne Connector Line interagiert mit keinem anderen Port und ist somit überflüssig.

```
1.     // Checks if all Ports of a given List of ProxyPorts do have at least one ConnectorLine
 2.     // Returns True if all Ports have at least one ConnectorLine
 3.     // Returns False if one Ports does not have a ConnectorLine
 4.     public boolean checkIfAllProxyPortsAreConnected(List<PresentationElement> listProxyPorts)
{
 5.         for(int i = 0; i < listProxyPorts.size(); i++){
 6.             var pathElements = getAllPathsFromPresentationElement(listProxyPorts.get(i));
 7.
 8.             if(pathElements.size() == 0)
 9.                 return false;
10.         }
11.         return true;
12.     }
```

Wie bei den BDDs auch, ist eine Überladung von Eingangs- und Ausgangs-Schnittstellen innerhalb eines Diagramms unvorteilhaft. In Abbildung 11 wird, genau wie in Abbildung 4, die Funktion countProxyInputAndOutput angewendet. Die Component_1 wurde um die funktionalen Schnittstellen des suD_WorstCase erweitert, wie das die Dekompositionsregeln vorschreiben würden. Das Plugin liefert genau wie bei den BDDs die Zwei Smells, dass die Anzahl Input Ports Zwei überstiegen hat und die Menge an Ausgangsports größer als Eins ist.

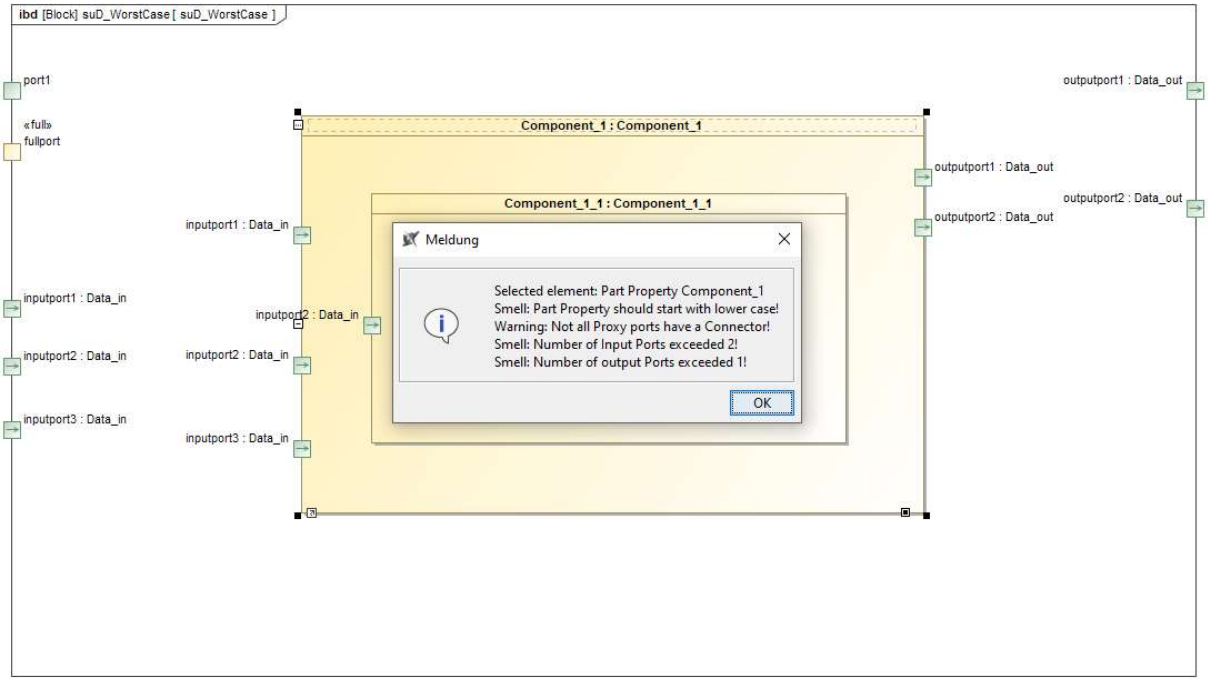

Abbildung 11: Zu viele Input-, Output-Ports am Part.

Natürlich ist es auch wichtig zu bestimmen, was für Ports in einem IBD genutzt wurden. So gelten für Ports in IBDs die gleichen Regeln, wie für Ports in BDDs. Vorzugsweise sollten nur Proxy Ports genutzt werden. Sollte ein anderer Typ von Port genutzt werden, so sollte ausschließlich diese Art verwendet werden. Das Mischen von Porttypen ist keine gute Idee. Wie an Abbildung 3 verdeutlicht wurde, wird in Abbildung 12 der gleiche Code genutzt. Der Component_1 wurde ein neuer Port „inputport4“ hinzugefügt. Bei diesem Port handelt es sich um einen normalen Port. Durch das Mischen zweier Porttypen wird der Meldung ein Error hinzugefügt, zusätzlich ist durch den Verzicht auf ausschließliches Nutzen von Proxy Ports ein Smell entstanden.

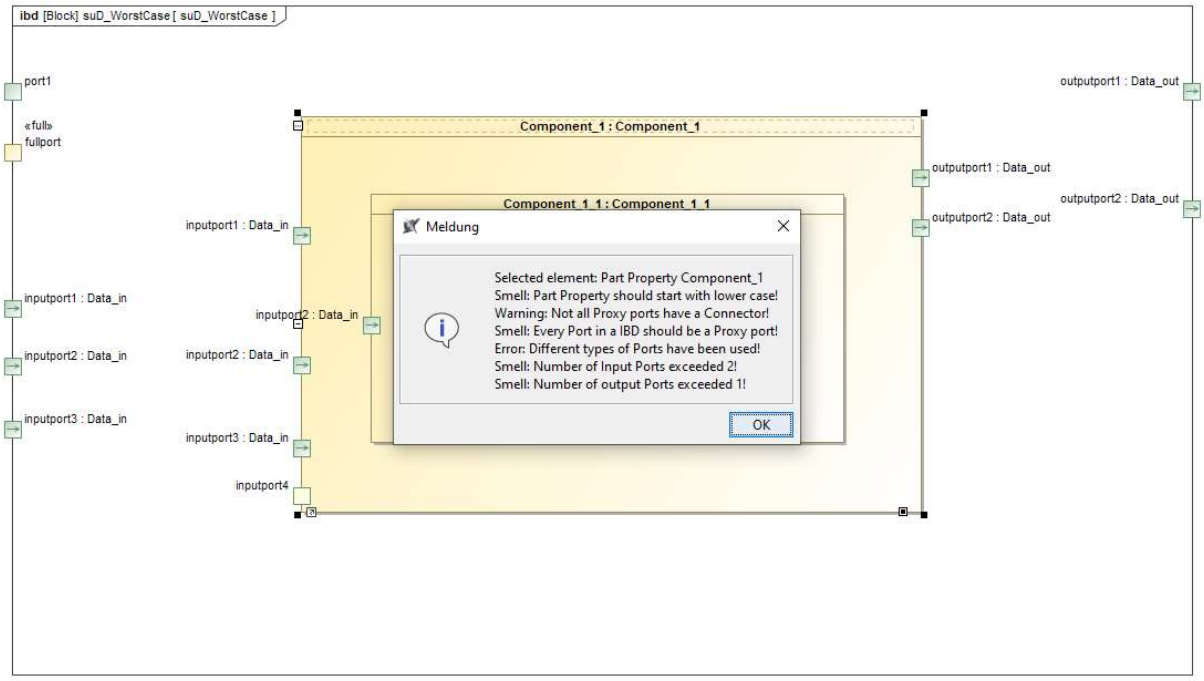

Abbildung 12: Es sind nicht nur Proxy Ports vorhanden, es wurde in einem IBD verschiedene Arten von Ports verwendet.

Nach dem Gesetz von Demeter, sollten Blöcke nur miteinander kommunizieren, wenn sie direkte Nachbarn sind. Die Funktion „checkIfConnectorsDoNotSkipPartProperties“ bedient sich aller Ports des übergebenen Blockes und überprüft, über die Connector Lines der Ports, ob Parts miteinander kommunizieren die nicht über einer direkten Beziehung miteinander verbunden sind.

```
 1.     // Checks if Connectors of all the ProxyPorts from a given presentation Element Skip Part
Properties on its way
 2.     // Returns True if no Connectors skip Part Properties
 3.     // Returns False if at least one Connector skips a Part Property
 4.     public boolean checkIfConnectorsDoNotSkipPartProperties(PresentationElement
presentationElement) {
 5.         var element = presentationElement.getElement();
 6.         var elementName = element.getHumanName().split(" ");
 7.         var parent = element.getOwner();
 8.         var parentName = parent.getHumanName().split(" ");
 9.         var proxyPorts = getAllProxyPortsFromPresentationElement(presentationElement);
10.
11.         for(var proxyPort : proxyPorts){
12.             var paths = getAllPathsFromPresentationElement(proxyPort);
13.
14.             for(var path : paths) {
15.                 var supplierBlockName =
path.getSupplier().getElement().getOwner().getHumanName().split(" ");
16.                 var clientBlockName =
path.getClient().getElement().getOwner().getHumanName().split(" ");
17.
18.                 //Supplier is selected Element
19.                 if(supplierBlockName[supplierBlockName.length -
1].equals(elementName[elementName.length - 1])) {
20.                     var parentClientName =
path.getClient().getParent().getParent().getHumanName().split(" ");
21.
22.                     //Parent of Client is the same of the selected Element (Feature
Interaction)
23.                     if(parentName[parentName.length -
1].equals(parentClientName[parentClientName.length - 1])){
24.                         // do nothing
25.                     }
26.                     //Client is not Parent (Skipped Part Property)
27.                     else if(!clientBlockName[clientBlockName.length -
1].equals(parentName[parentName.length - 1]))
28.                         return false;
29.                 }
30.                 //Client is selected Element
31.                 else {
32.                     var parentSupplierName =
path.getSupplier().getParent().getParent().getHumanName().split(" ");
33.
34.                     //Parent of Supplier is the same of the selected Element (Feature
Interaction)
35.                     if(parentName[parentName.length -
1].equals(parentSupplierName[parentSupplierName.length - 1])){
36.                         // do nothing
37.                     }
38.                     //Supplier is not Parent (Skipped Part Property)
39.                     else if(!supplierBlockName[supplierBlockName.length -
1].equals(parentName[parentName.length - 1]))
40.                         return false;
41.                 }
42.             }
43.         }
44.         return true;
45.     }
```

Um den Nutzen der Funktion zu veranschaulichen, wurde der Proxy Port inputport2 der Component_1_1 über eine Connector Line mit dem inputport2 des suD_WorstCase Blocks verbunden. Mit dem Plugin wurde die Component_1_1 ausgewählt und eine Meldung erzeugt. Die Meldung beinhaltet nun einen Error, da Component_1_1 mit dem suD_WorstCase kommuniziert aber dazwischen noch die Component_1 liegt.

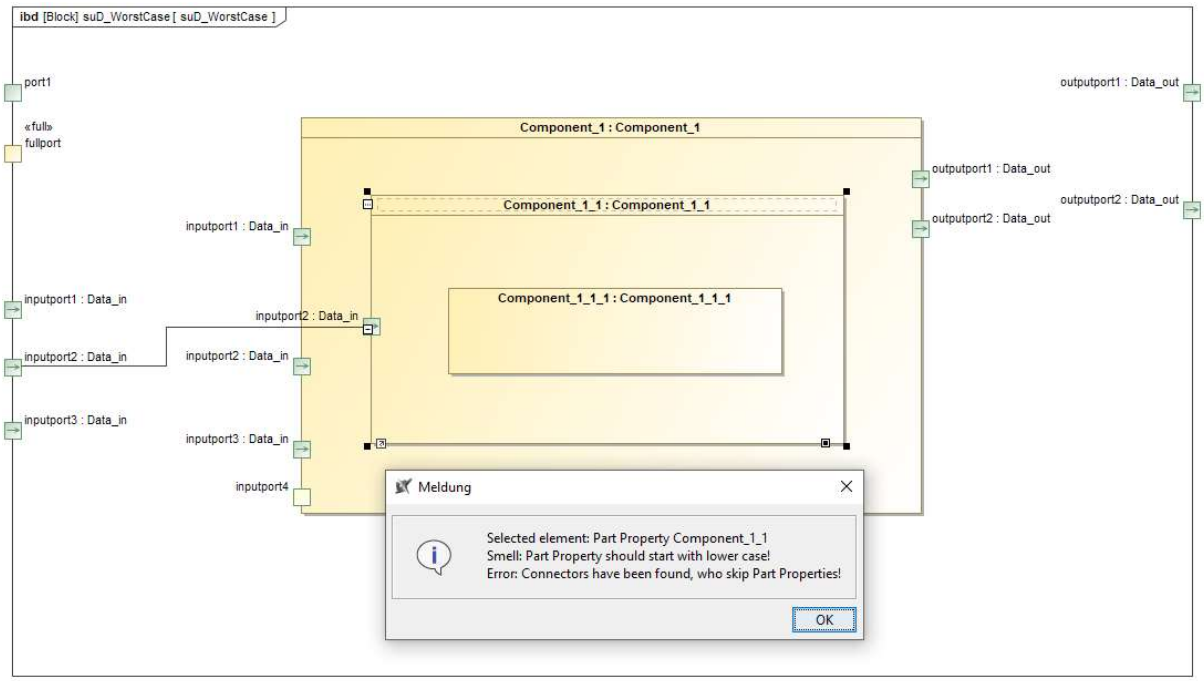

Abbildung 13: Parts Kommunizieren mit nicht direkten Nachbarn

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung der Ergebnisse

Um zu untersuchen, wie sich die Qualität von Strukturdiagrammen der SysML, in Bezug auf die Semantik und Syntax, bewerten lässt und wie diese Bewertung automatisiert werden kann, wurden zuerst, mit Hilfe einer qualitativen Forschung, diverse Bewertungskriterien erhoben und analysiert. Mit den gesammelten Bewertungskriterien wurde dann ein Plugin für den Cameo Systems Modeler entwickelt, welches ein Strukturdiagramm anhand dieser Kriterien bewertet und den Nutzer eine Meldung über die Qualität des Strukturdiagrammes liefert. Die Bewertungskriterien wurden zum Teil aus etablierten Prinzipien, wie das Gesetz von Demeter, abgeleitet und andere wurden angesehenen Experten entnommen.

Die Rückmeldung des Plugins liefert eine Bewertung zu dem jeweiligen Kriterium. Diese kann genutzt werden, um mit Hilfe einer Nutzwertanalyse Das Diagramm zu bewerten. So kann ein Smell, als ein Multiplikator von 0,8, mit der Gewichtung verrechnet werden. Keine Rückmeldung kann als 1 betrachtet werden, eine Warning als 0,5 und ein Error als eine 0. Das Aufsummieren der „Punkte“ kann dann als eine Einschätzung, über die Qualität des Diagramms, dienen. Je höher die Punktzahl, desto mehr Bewertungskriterien wurden eingehalten.

Da sich die Relevanz der Qualitätsmerkmale an die Diagramme, wie die Übersicht oder ein geringes Risiko auf Verwechslungsgefahr, von Projekt zu Projekt ändert, ist auch die Gewichtung der Bewertungskriterien anzupassen. Die hier präsentierten Gewichtungen stellen lediglich eine grobe Verallgemeinerung dar und entstammen meinen eigenen Erfahrungen und Kenntnissen.

Bewertungskriterium für ein BDD	Gewichtung
Die Tiefe der Composition-Associations ist so gering wie möglich.	15
Zwei Blöcke Kommunizieren nur dann miteinander, wenn sie direkte Nachbarn sind.	10
Die Anzahl der Ports für die Kommunikation zwischen Blöcken ist so gering wie möglich.	15
Die Anzahl an Input-Ports des System under Development ist so gering wie möglich.	5
Die Anzahl an Output-Ports des System under Development ist so gering wie möglich.	5

Blöcke starten mit einem UpperCamelCase.	5
Properties von Blöcken starten mit einem lowerCase.	5
Properties von Blöcken sind nicht identisch nach dem Namen des definierenden Blockes benannt.	5
Die Dekomposition ist korrekt.	20
Die Diagrammelemente sind entsprechend ihrer Dekompositionshierarchie positioniert.	10
Jeder Block wird durch ein eigenes IBD definiert.	5

Tabelle 1: Bewertungskriterien für Block Definition Diagramme

Bewertungskriterium für ein IBD	Gewichtung
Die Anzahl an Input-Ports eines Parts ist so gering wie möglich.	5
Die Anzahl an Output-Ports eines Parts ist so gering wie möglich.	5
Alle Parts, des zu beschreibenden Blockes, sind abgebildet.	15
Redundanz bei der Portbeschreibung ist so gering wie möglich.	5
Keine überlappenden Connector Lines sind vorhanden.	5
Parts sind entsprechend ihrer Zugehörigkeit positioniert.	15
Nur eine Art von Port sollte genutzt werden, vorzugsweise Proxy Ports.	10
Alle Proxy Ports sind, durch ein Interface Block, typisiert.	10
Alle Proxy Ports besitzen durch ihren typisierenden Block ein Flow Property.	15
Jeder Port ist Teil mindestens einer Connector Line.	15

Tabelle 2: Bewertungskriterien für Interne Block Diagramme

Bewertungskriterium für Paket Diagramme	Gewichtung
Ein Top-Level Paket Diagramm ist vorhanden.	15
Ein Paket Diagramm ist vorhanden, welches das Modell mittels Unterpaketen strukturiert	40
Die Modellhierarchie wurde mittels Organisationsprinzipien strukturiert.	10
Das Paket Diagramm ist mittels Containments strukturiert.	15
Packages sollten nach ihrem Inhalt oder ihrer Funktionalität benannt werden.	5
Alle Pakete besitzen einen einzigartigen Namen innerhalb ihres Namespaces.	15

Tabelle 3: Bewertungskriterien für Paket Diagramme

Das Plugin verwendet Funktionen, um das Diagramm anhand der Bewertungskriterien zu analysieren. Die Funktionen Melden eine Einstufung des Bewertungsmaßstabes und legen, zusammen mit der Gewichtung des jeweiligen Bewertungskriteriums, somit die Qualität des Diagramms fest.

Funktion	Erklärung
checkIfPresentationElementStartsWithLowerCase	Überprüft ein Presentation Element, ob es mit einem kleinen Buchstaben beginnt.
checkIfAllProxyPortsHaveAType	Überprüft alle Proxy Ports auf Typisierung.
getAllProxyPortsFromPresentationElement	Speichert alle Proxy Ports, eines Presentation Elements, in eine Liste.
getAllFullPortsFromPresentationElement	Speichert alle Full Ports, eines Presentation Elements, in eine Liste.
getAllPortsFromPresentationElement	Speichert alle Ports, eines Presentation Elements, in eine Liste.
countProxyInputAndOutput	Zählt die Eingangs Ports, Ausgangs Ports und Ports ohne eine Richtung einer übergebenen Liste mit Proxy Ports.
checkIfDecompositionIsDoneRightInAllAboveBlocks	Wiederholt die Anwendung der checkIfDecompositionIsDoneRightInBlock

	Above Funktion, auf alle Eltern Blöcke des übergebenen Presentation Elements.
checkIfDecompositionIsDoneRightInBlockAbove	Überprüft die korrekte Dekomposition der gefundenen Functional Interfaces, des übergebenen Blockes an die Funktion.
getNestingDepthFirstSearch	Berechnet die Tiefe der Dekompositionshierarchie.
checkIfSuDHasADiagram	Überprüft, ob das SuD, welches zu dem an die Funktion übergebenen Blocks gehört, ein eigenes IBD besitzt.
checkIfAllProxyPortsAreConnected	Überprüft, ob jeder Port, des übergebenen Presentation Elements, Teil einer Connector Line ist.
checkIfConnectorsDoNotSkipPartProperties	Überprüft, ob der übergebene Block nur mit seinen direkten Nachbarn kommuniziert.

Tabelle 4: Funktionen des Plugins

5.2 Interpretation der Ergebnisse

Welche Schlussfolgerungen können, mit Hilfe der Ergebnisse dieser Forschung, gezogen werden?

Wie erwartet, zeigt die Menge an erhobenen Bewertungskriterien, dass sich Diagramme sowohl objektiv wie auch subjektiv Bewerten lassen. Mit der Entwicklung eines Plugins für den Cameo Systems Modeler zeigte sich zusätzlich, dass sich viele der Kriterien an die Strukturdiagramme auch automatisiert Messen und Bewerten lassen, aber nicht alle.

Durch das Zusammentragen und zur Verfügung stellen einer Liste mit hochwertigen Heuristiken, Good Practices und Prinzipen, ist diese Forschung einen Schritt nähergekommen, eine weitreichende Quelle von Modellierungsprinzipien für Anwendern der SysML zu schaffen.

Mithilfe des entwickelten Plugins ist es möglich ein Diagramm mithilfe einer Nutzwertanalyse, auf einer wissenschaftlichen Weise und automatisiert, zu analysieren und alternative Diagramme in Betracht ziehen zu können.

Durch das Bestimmen der Bewertungskriterien ist es auch erst möglich, festzulegen was Qualität bei Diagrammen überhaupt bedeutet. Denn um eine positive oder negative Änderung feststellen zu können, muss erst eine Grundlage geschaffen werden, nach der man sich richten kann. Diese Forschung konnte folgende Eigenschaften für hochwertige Diagramme festlegen: Übersicht, Flexibilität, geringe Fehleranfälligkeit, korrekte Syntax und Semantik, keine Missverständnisse können entstehen, keine Verwechslungsgefahr ist vorhanden und die Redundanz ist so gering wie möglich. Diese Erkenntnisse helfen dem Modellierer zusätzlich bei der Entscheidung, welche Eigenschaften eines Diagramms im Modellierungsprozess priorisiert werden sollten und aus welchem Grund.

Auch konnte durch die qualitative Forschung gezeigt werden, dass Heuristiken und Good Practices anderer Themengebiete, wie der objektorientierten Programmierung, durchaus auch auf das Modellieren von Diagrammen angewendet werden können.

5.3 Begrenzungen dieser Forschung

Während der Durchführung ist die Forschung auch auf einige Limitationen gestoßen.

Das Plugin legt bis jetzt nur die Bewertung der Kriterien fest, die eigentliche Nutzwertanalyse wurde noch nicht automatisiert. Das liegt daran, dass es wenig Sinn macht die Bewertung zu Gewichten, wenn sich die Prioritäten an die Diagrammeigenschaften von Projekt zu Projekt und von Unternehmen zu Unternehmen unterscheiden. Die angegebenen Gewichtungen der hier präsentierten Bewertungskriterien, bilden somit nur eine grobe Verallgemeinerung und sollten für jedes Projekt individuell angepasst werden.

Bei der Bewertung von Diagrammen sind auch Grenzen sichtbar geworden, so lässt sich die Qualität von Diagrammen zwar Bewerten, aber nur einige Kriterien sind tatsächlich auf alle Diagramme anwendbar. So sind zum Beispiel Kriterien an die Diagrammstruktur oder Namenskonventionen nur sehr schwer zu vereinheitlichen und sollten für jedes Modell oder Projekt angepasst werden.

Gezeigt hat sich auch, dass wenige Informationen vorhanden sind, die einem den Umgang mit der MagicDraw API beibringen, zudem ist die Dokumentation nicht detailliert genug und Einsteiger unfreundlich. Dementsprechend ist das Plugin nicht optimiert und die Entwicklung hat sich in die Länge gezogen.

Ein weiteres Manko ist, dass das Plugin nur mit der „MagicDraw modeling platform“ oder ihren Ablegern wie Cameo Systems Modeler genutzt werden kann. Nutzer anderer SysML Werkzeuge müssen, sofern vorhanden, das Plugin an die anderen APIs anpassen.

5.4 Empfehlungen für weiterführende Forschungen

Aufgrund der Limitationen sind einige Ansätze für weiterführende Forschungen ersichtlich geworden.

Wie bereits erwähnt, bewertet das Plugin lediglich die einzelnen Bewertungskriterien, und setzt diese nicht in Verbindung mit der Gewichtung. Eine Erweiterung des Plugins um ein Graphisches User Interface, welches ermöglichen soll, die Gewichtung im Tool selbst vorzunehmen, wäre eine Empfehlung für zukünftige Forschung. Dies ermöglicht die vom Plugin bestimmte Bewertung mit der Gewichtung verrechnen zu können und dadurch den Nutzwert des Diagramms, automatisiert und flexibel, zu bestimmen.

Passend dazu bietet es sich an Forschung zu betreiben, die die Verteilung der Gewichtungen anhand einer mathematischen oder wissenschaftlichen Basis bestimmt und nicht aufgrund von persönlichen Erfahrungen und Kenntnissen.

Forschung sollte auch für die MagicDraw API Dokumentation betrieben werden. Diese ist nicht detailliert genug und gibt wenig Hilfestellungen. Aufgrund der mangelnden zeit und unübersichtlichen Dokumentation ist das Plugin zudem nicht ordentlich optimiert und kann weiterhin verbessert werden.

Natürlich kann das Plugin auch um neue Funktionalitäten erweitert werden. Dafür müssen allerdings mehr Bewertungskriterien erhoben werden, was wiederum mehr Forschung voraussetzt.

Ein weiteres Problem war das Implementieren der Überprüfung von Namenskonventionen und der Diagrammstrukturen. Diese können nämlich auf viele verschiedene Arten umgesetzt werden und jede Methode bietet seine eigenen Vor- und Nachteile. Eine Empfehlung für zukünftige Forschung ist, dass fördern eines Standards für die Strukturierung von Diagrammen und Namenskonventionen.

Die Literatur Recherche hat außerdem deutlich gemacht, dass nur wenige Versuche getätigt wurden, eine Ansammlung von Modellierungsprinzipien zu erstellen. Das Bilden einer einheitlichen und für alle zugänglichen Quelle für Good Practices, Heuristiken und Prinzipien für die SysML, würde vielen Anwendern helfen und zukünftige Forschungen vereinfachen.

6 Fazit

Diese Arbeit konnte mithilfe einer qualitativen Forschungsmethode, eine Sammlung von Modellierungsprinzipen für die Strukturdiagramme BDD, IBD und PKG der SysML erstellen. Aus den Prinzipien wurden dann Bewertungskriterien abgeleitet, mit denen sich Diagramme auf verschiedene Aspekte hin Bewerten lassen. Weiter konnte durch das Implementieren eines Plugins für den Cameo Systems Modeler, mit Hilfe der MagicDraw API, eine Automatisierung der Bewertung von Strukturdiagrammen über die erhobenen Kriterien vorgenommen werden.

Diese Forschung konnte die Hypothese, dass sich die Qualität von Strukturdiagrammen der SysML bewerten lässt, belegen. Dadurch lässt sich die zu Anfangs gestellte Forschungsfrage folgendermaßen beantworten: Die Qualität der Semantik und Syntax von Strukturdiagrammen, lässt sich mithilfe der hier vorgestellten Bewertungskriterien Analysieren. Das Plugin, welches die Bewertung automatisiert, führt diese auf die Diagramme aus und liefert dem Nutzer eine Bewertung, die Auskunft gibt, inwiefern die Kriterien umgesetzt wurden.

Anhand der Ergebnisse konnten Schlussfolgerungen gezogen werden und neue Erkenntnisse konnten erlangt werden. So konnte in Erfahrung gebracht werden, dass einige Kriterien nicht so einfach implementiert werden können, besonders Namenskonventionen und die Analyse der Diagrammstrukturierung. Ersichtlich wurde auch, dass es wenige Quellen gibt, die eine Sammlung von Modellierungsprinzipen für die SysML, anbieten. Diese Arbeit konnte dem entgegenwirken, da die Prinzipien verschiedensten Experten entnommen wurden und in Verbindung gebracht wurden und somit ein Grundbaustein für eine einheitliche Quelle von Empfehlungen und Best Practices gelegt wurde. Auch konnte festgehalten werden, welche Eigenschaften Aussage über die Qualität eines Strukturdiagramms tätigen, wodurch der Modellierer dabei unterstützt wird, die Priorisierung der Qualitätskriterien Projektabhängig vorzunehmen. Zusätzlich konnte gezeigt werden, dass Heuristiken anderer Themenfelder in den Modellierungsprozess mit einfließen können.

Zum Abschluss werden noch einige Empfehlungen für zukünftige Forschung ausgesprochen. So lässt sich das Plugin noch um die Automatisierung der Nutzwertanalyse erweitern oder anders gesagt, dass Plugin kann um die Verrechnung der bereits Implementierten Bewertung mit der Gewichtung erweitert werden. Im gleichen Zuge wäre weitere Forschung bezüglich der Gewichtungen vorteilhaft, mit dem Grund, dass diese nach einer mathematischen und/oder wissenschaftlichen Basis festgelegt werden können. Auch bietet sich Forschung an, die sich das Standardisieren der Namenskonventionen und der Diagrammstrukturierung zur Aufgabe macht. Und zuletzt das offensichtlichste, mehr Bewertungskriterien zu erheben und das Plugin um diese zu Erweitern

Glossar

Anti Patterns
Methoden die für den Erfolg eines Projektes ungünstig oder schädlich für den Erfolg eines Projektes oder einer Organisation sind. (wikipedia, 2023) 8
Best Practice Patterns
Best mögliche Methode für Erfolg eines Projektes 8
Black Box
Als Black Box bezeichnet man in der Systemtheorie ein System, von welchem im gegebenen Zusammenhang nur das äußere Verhalten betrachtet werden soll. (wikipedia, 2023) 5
Composition-Associations
Beziehung zwischen einem Ganzen und seinen Teilen. 10
Connector Lines
Graphisches Verbindungsglied zweier Elemente 13
Containments
Graphische Eingrenzung von Elementen mittels Relationen zwischen Paketen.....15
Feature Interaction
Ein Software Engineering Konzept, wenn das Interagieren zweier Features das Verhalten eines oder beider Features verändern würde (wikipedia, 2023).....12
Flow Property
Eigenschaft eines Blockes, welche ein einzelnes Fluss Element von/zu einem Block mit einer Richtungsangabe signalisiert. 14
Fork
Ein Node, welches Objekt- oder Datenfluss eines Nodes an mehrere weitergibt. 13
Good Practices
Praktisch erfolgreiche Lösungen oder Verfahrensweisen, auch auf längere Sicht und in einer Gesamtschau aller Belange. (good_practice.htm, 2023) 2
Heuristiken
Methoden mit begrenstem Wissen und wenig Zeit, die dennoch zu wahrscheinlichen Aussagen oder zu praktikablen Lösungen kommen. (wikipedia, 2023)..... 3
Junction
Ein Node, welches den Objektfluss oder Datenfluss zusammenführt und an einen Node weitergibt.....13
Lifecycle
Produkt Lebenszyklus, von einführung bis es vom Markt entfernt wird..... 1
Namespace
Namespaces werden verwendet, um Code in logischen Gruppen zu organisieren und Namenskonflikte zu vermeiden..... 6
Nesting
Nesting ist das verschachteln von Informationen oder Objekten. 13
Node
Symbole für die modellierung von Aktivitätsdiagrammen 13
Operationalen Kontext
Personen, Prozesse, Externe Systeme die aus der Umgebung des Systems über Schnittstellen auf das System zur Laufzeit zugreifen. 11
Parts
Ein Block, welches über eine starke Abhängigkeitsbeziehung zwischen dem Ganzen (Block) und seinen Bestandteilen (Parts) definiert ist.....12
Ports
Schnittstellenteil für atomare Interaktion zwischen System und Umgebung. 11
Properties

Block zugehörige Eigenschaften. Ein Einzelteil (Part), eine Referenz oder ein Wert. 11
Schnittstelle
Gemeinsamer Anteil von System und Umgebung. 11
Summed-Flow
Zummanführung mehrere Kontrollflüsse oder Objektflüsse 13
System under Development
Das durch das Diagramm beschriebene (Teil-)System. 10
Top-Level
Oberste Stufe der Projekthierarchie, unterhalb des Projektwurzelverzeichnisses. 15
White Box
Als White Box bezeichnet man in der Systemtheorie ein (möglicherweise sehr komplexes) System, von welchem im gegebenen Zusammenhang nur das innere Verhalten betrachtet werden soll. 6

Literaturverzeichnis

Campo, K. X. (2022). Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature. *Systems Engineering Volume 26*.

Darren, K. R. (2016, 8 1). Model-Based Systems Engineering and Software Engineering resume.

Darren, K. R. (2023, 5 11). *DO NOT use Property names that are identical to the names of the Classifier (Class, DataType, Block, ValueType) that type them!* Retrieved from webel.com.au: https://www.webel.com.au/node/1405

Darren, K. R. (2023, 05 14). *If instances of Blocks "communicate" with each other use an Association between them.* Retrieved from webel.com.au: https://www.webel.com.au/node/1930

Darren, K. R. (2023, 05 15). *MagicDraw/Cameo: Keep most elements out of the top-level of a project (i.e. not directly under the project Root). Have one top-level index/content/package diagram.* Retrieved from webel.com.au: https://www.webel.com.au/node/2264

Darren, K. R. (2023, 05 12). *UML/SysML: In Internal Block Diagrams: Consider hiding the name of a named Port or Property in a Diagram if its Type is sufficient to indicate its role.* Retrieved from webel.com.au: https://www.webel.com.au/node/1565

Darren, K. R. (2023, 05 12). *UML/SysML: In Internal Block Diagrams: If you have a Port with a name that indicates a unique role AND and if there is an ItemFlow on a Connector that implies or suggests the Type of the Port, consider hiding the Type on the Port symbol.* Retrieved from webel.com.au: https://www.webel.com.au/node/3194

Darren, K. R. (2023, 5 11). *Use 'UpperCamelCase' (a.k.a. PascalCase) names for Classifiers such as UML Classes and SysML Blocks) to avoid confusion with 'lowerCase' Property names [see however special naming conventions for acronyms and SysML contract and flow Blocks]*. Retrieved from webel.com.au: https://www.webel.com.au/node/669

Darren, K. R. (2023, 05 13). *Webel Best Practice: SysML/SysPhS: DO NOT use overlapping Connector lines from/to one Port (can be misunderstood as "summed" flow and/or physical node/fork/junction).* Retrieved from webel.com.au: https://www.webel.com.au/node/2779

Darren, K. R. (2023, 05 11). *Webel's Best Practice policy notes for UML and SysML1.x and the MagicDraw/Cameo tools*. Retrieved from webel.com.au: https://www.webel.com.au/node/668

Deutsche Akademie der Technikwissenschaften (acatech). (2020). *Industrie 4.0 Maturity Index. Die digitale Transformation von Unternehmen gestalten - UPDATE 2020 - (acatech STUDIE).*

Friedenthal, S. (2005). *A Practical Guide to SysML: The Systems Modelling Language.* Elsevier.

Friedenthal, S., Dori, D., & Mordecai, Y. (2023, 04 27). *Representing_Systems_with_Models*. Retrieved from sebokwiki.org: https://sebokwiki.org/wiki/Representing_Systems_with_Models

Friedenthal, S., Moore, A., & Steiner, R. (2015). Book Organization. In *A Practical Guide to SysML* (pp. xvii-xviii).

Friedenthal, S., Moore, A., & Steiner, R. (2015). Diagram Layout. In *A Practical Guide to SysML* (p. 96).

Friedenthal, S., Moore, A., & Steiner, R. (2015). Establishing Model Quality Criteria. In *A Practical Guide to SysML* (p. 23).

Friedenthal, S., Moore, A., & Steiner, R. (2015). Organizing a Package Hierarchy. In *A Practical Guide to SysML* (p. 104).

Friedenthal, S., Moore, A., & Steiner, R. (2015). Organizing a Package Hierarchy. In *A Practical Guide to SysML* (p. 105).

Friedenthal, S., Moore, A., & Steiner, R. (2015). Packages as Namespaces. In *A Practical Guide to SysML* (p. 107).

Geschichte - Systems Engineering. (2023, 4 30). Retrieved from lxjkh: https://de.lxjkh.com/info_detail/?de_3-1627669002-1&Geschichte_Systems_Engineering

good_practice.htm. (2023, 05 29). Retrieved from olev.de: https://olev.de/g/good_practice.htm

Hein, A. M., Weilkiens, T., Karban, R., & Zamparelli, M. (2011, January). Cookbook for MBSE with SysML.

Highsmith, J. (2023, 05 10). */web/20120910145759/http://agilemanifesto.org/history.html*. Retrieved from web.archive.org: https://web.archive.org/web/20120910145759/http://agilemanifesto.org/history.html

International Council on Systems Engineering (INCOSE). (2007). *Systems Engineering Vision 2020.*

International Council on Systems Engineering (INCOSE). (2014). *Systems Engineering Vision 2025.*

International Council on Systems Engineering (INCOSE). (2022). *Systems Engineering VIsion 2035.*

International Council on Systems Engineering (INCOSE). (2023, 04 27). *systems-engineering*. Retrieved from incose: https://www.incose.org/systems-engineering

Keep It Short and Simple. (1938). *The Minneapolis Star*.

Lieberherr, K. L., Holland, I., & Riel, A. J. (1988). Object-Oriented Programming: An Objective Sense of Style. In *oopsla* (pp. 323-334). San Diego, CA.

Martin, R. C. (2008). Function Arguments. In *Clean Code: A Handbook of Agile Software Craftsmanship* (pp. 40-41). Prentice Hall Pearson Education.

Martin, R. C. (2008). Too much Information. In *Clean Code: A Handbook of Agile Software Craftsmanship* (pp. 291-292). Prentice Hall Pearson Education.

Martin, R. C. (2008b). *Clean Code: A Handbook of Agile Software Craftsmanship.*

National Academy of Engineering. (2023, 04 30). *challenges*. Retrieved from engineeringchallenges: http://www.engineeringchallenges.org/challenges.aspx

Object Management Group. (2019, 11). OMG Systems Modeling Language (OMG SysML) Version 1.6.

Object Management Group. (2023, 04 27). *what-is-sysml.htm*. Retrieved from omgsysml: https://www.omgsysml.org/what-is-sysml.htm

Rich, B. R. (1995). Clarence Leonard (Kelly) Johnson 1910-1990: A Biographical Memoir . Washington, D.C, United States of America: National Academies Press.

SysML. (2023, 04 27). *what-is-block-definition-diagram.html*. Retrieved from syml.org: https://sysml.org/sysml-faq/what-is-block-definition-diagram.html

SysML. (2023, 04 27). *what-is-internal-block-diagram.html*. Retrieved from sysml.org: https://sysml.org/sysml-faq/what-is-internal-block-diagram.html

SysML. (2023, 04 27). *what-is-package-diagram.html*. Retrieved from sysml.org: https://sysml.org/sysml-faq/what-is-package-diagram.html

SysML Partners. (2023, 5 12). *SysML Open Source Project: What is SysML? Who created SysML?* Retrieved from sysml.org: https://sysml.org/

Systems Engineering Body of Knowledge (SEBoK). (2023, 4 30). *sebokwiki*. Retrieved from Model-Based_Systems_Engineering_Adoption_Trends_2009-2018: https://sebokwiki.org/wiki/Model-Based_Systems_Engineering_Adoption_Trends_2009-2018

Towers, J., & Hazle, A. (2020, November). Good Practice in MBSE Model Verification and Validation.

Weilkiens, T. (2023, 05 14). *SysML Full Ports versus Proxy Ports*. Retrieved from mbse4u.com: https://mbse4u.com/2013/09/23/sysml-full-ports-versus-proxy-ports/

Weilkiens, T., & Muggeo, C. (2023, 05 14). *The MBSE Podcast*. Retrieved from mbse-podcast.rocks: https://mbse-podcast.rocks/

wikipedia. (2023, 05 29). *Anti-Pattern*. Retrieved from de.wikipedia.org: https://de.wikipedia.org/wiki/Anti-Pattern

wikipedia. (2023, 05 29). *Black_Box_(Systemtheorie)*. Retrieved from de.wikipedia.org: https://de.wikipedia.org/wiki/Black_Box_(Systemtheorie)

wikipedia. (2023, 05 29). *Feature_interaction_problem*. Retrieved from en.wikipedia.org/: https://en.wikipedia.org/wiki/Feature_interaction_problem

wikipedia. (2023, 05 29). *Heuristik*. Retrieved from de.wikipedia.org: https://de.wikipedia.org/wiki/Heuristik

Anhänge

Anhang 1: Excel Tabelle mit Bewertungskriterien

https://github.com/MariusHeinrichs/Abschlussarbeit-Fh-Aachen-FB5/blob/main/Bachelorarbeit_Bewertungskriterien.xlsx

Anhang 2: Plugin Source Code

https://github.com/MariusHeinrichs/Abschlussarbeit-Fh-Aachen-FB5/tree/main/My%20Plug-in%201

Anhang 3: Erklärung und Geheimhaltung

https://github.com/MariusHeinrichs/Abschlussarbeit-Fh-Aachen-FB5/blob/main/Bachelorarbeit_Erklaerung_und_Geheimhaltung_Unterschrieben.pdf

Erklärung

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachwels versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, im Monat Jahr (vollständige, handschriftliche Unterschrift)

05 2023 Heinrichs

Geheimhaltung

Diese Bachelorarbeit darf weder vollständig noch auszugsweise ohne schriftliche Zustimmung des Autors, des betreuenden Referenten bzw. der Firma _____ vervielfältigt, veröffentlicht oder Dritten zugänglich gemacht werden.

Wahlweiser Zusatz:
Von der Geheimhaltung ausgenommen ist ausschließlich die Kurzfassung zur "Darstellung im Internet" auf Seite ___ der Arbeit .