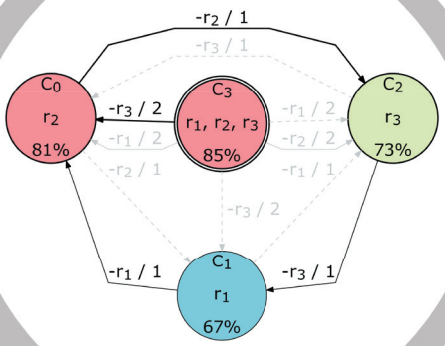
The Karlsruhe Series on
Software Design
and Quality
28

Entwurfsoptimierung von selbst-adaptiven Wartungsmechanismen für software-intensive technische Systeme

Lukas Märtin



Scientific Publishing

Lukas Märtin

Entwurfsoptimierung von selbst-adaptiven Wartungsmechanismen für software-intensive technische Systeme

The Karlsruhe Series on Software Design and Quality
Volume 28

Chair Software Design and Quality
Faculty of Computer Science
Karlsruhe Institute of Technology

and

Software Engineering Division
Research Center for Information Technology (FZI), Karlsruhe

Editor: Prof. Dr. Ralf Reussner

Entwurfsoptimierung von selbst-adaptiven Wartungsmechanismen für software-intensive technische Systeme

von
Lukas Märtin

KIT Scientific Publishing

Dissertation, Karlsruher Institut für Technologie
KIT-Fakultät für Informatik

Tag der mündlichen Prüfung: 11. Juli 2018
Gutachter: Prof. Dr. rer. nat. Ralf H. Reussner
Prof. Dr.-Ing. Ina Schaefer

Impressum

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2019 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1867-0067
ISBN 978-3-7315-0852-6
DOI: 10.5445/KSP/1000086097

Entwurfsoptimierung von selbstadaptiven Wartungsmechanismen für software-intensive technische Systeme

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

Lukas Märtin

aus Burgdorf (Niedersachsen)

Tag der mündlichen Prüfung: 11. Juli 2018
Erster Gutachter: Prof. Dr. rer. nat. Ralf H. Reussner
Zweite Gutachterin: Prof. Dr.-Ing. Ina Schaefer

Zusammenfassung

In vielen Anwendungsdomänen sind software-intensive technische Systeme potenziell schädlichen äußerlichen Einflüssen ausgesetzt und zugleich für die dadurch verursachte Wartung häufig unzugänglich. Für einen unterbrechungsfreien Betrieb ohne direkten Zugriff, sind ein Höchstmaß an inhärenter Fehlertoleranz in der Hardwareplattform und ein adaptives Selbstmanagement im zugrundeliegenden Softwaresystem nötig. Gelingt es, Fehlertoleranzen und selbst-adaptive Wartungsmechanismen bereits in frühen Entwurfsphasen umfassend vorzubereiten und einzubetten, können erhebliche Kostenersparnisse durch entsprechend ausgelegte Rekonfigurationsstrategien erreicht werden.

In heutigen Systemen werden Hardwareressourcen in homogener redundanter Ausprägung im Systementwurf vorgesehen und mittels etablierter Fehlerbehandlungsstrategien im Softwaresystem verwaltet. Auf dieser Basis unterstützen fehlertolerante Systeme eine autarke Erkennung, Lokalisierung und Isolation erwarteter Ressourcenfehler und teilweise auch deren autonome Kompensation. Trotz hochentwickelter Technologien unterliegt die selbst-adaptive Wartung natürlichen Limitierungen, die die Beteiligung eines Entscheidungsträgers unabdingbar machen. Dies trifft insbesondere dann zu, wenn neben der korrekten Funktionalität eine optimierte Systemgüte als wesentlich für einen erfolgreichen und langfristigen Betrieb gilt. Sobald notwendige Modifikationen über homogene Redundanzwechsel hinausgehen und die Stabilität der Qualität nicht garantiert werden kann, verbleibt als sichere Option oftmals nur der Wechsel in einen ressourcensparenden Notbetrieb, der erst wieder nach manuellen externen Anpassungen verlassen werden kann. Je nach Ausprägung der Änderungen führen die notwendigen Wartungsaktivitäten zu hohen personellen Aufwänden in der Entscheidungsfindung bei zugleich begrenztem Diagnosezugriff. Dies wiederum erhöht neben den Wartungskosten, ebenfalls die Ausfallzeiten sowie den Kommunikationsbedarf zwischen System und Wartungspersonal.

Aufbauend auf dem Stand der Technik werden in dieser Arbeit Konzepte entwickelt und erprobt, bei denen bereits in der Entscheidungsfindung je nach Grad der Redundanz und Variabilität im System, ein exponentiell großer Entwurfsraum möglicher alternativer Konfigurationen mit wechselseitigen Bedingungen und gegenläufigen Qualitätsattributen effizient untersucht wird. Grundsätzlich wäre dabei erschöpfend jede Konfiguration zu erheben, zu bewerten und nach verfügbaren Ressourcen zur Inbetriebnahme gegenüber Alternativen abzuwägen. Zur Anwendung bei limitierten Rechenzeiten werden in dieser Arbeit neuartige Konzepte zur effizienten Entscheidungsunterstützung in der Rekonfiguration software-intensiver technischer Systeme mit beschränktem Wartungszugriff vorgestellt. Bei der Umsetzung dieser Konzepte werden relevante technische und wirtschaftliche Effekte berücksichtigt.

Die Arbeit behandelt in diesem Zusammenhang die Frage nach dem Grad der Vorausberechenbarkeit adäquater Rekonfigurationen zur Reduktion von Wartungsaufwänden. Dabei folgt die Arbeit der Idee

- effektive Konfigurationsoptionen zur Entwurfszeit zu identifizieren,
- diese qualitativ abzustufen und
- zur effizienten Entscheidungsfindung in Entwicklung und Betrieb zu verknüpfen.

Eine frühzeitige architekturzentrierte Auswirkungsanalyse von Rekonfigurationen unterstützt die Fehlerbehandlung in Entwicklung und Betrieb somit.

Entgegen rein redundanzorientierter Ansätze, basiert die Fehlerbehandlung im Ansatz dieser Arbeit auf der prädiktiven Vorausberechnung adäquater Konfigurationsalternativen im relevanten Lösungsraum. Das Wissen über Konfigurationsbeziehungen wird im Entwurf explizit sichtbar gemacht und für Entscheidungen zur Laufzeit aufbereitet und manifestiert. Das resultierende Expertensystem ermittelt autark und kosteneffizient eine Alternative und stellt damit Entscheidungsträgern die Anpassungsmöglichkeiten transparent dar. Der Kommunikationsbedarf und der personelle Aufwand in der Wartung software-intensiver technischer Systeme wird somit deutlich reduziert. Die strukturierte und geführte Exploration von Alternativen fokussiert sich dabei nicht ausschließlich auf einen Ressourcenaustausch innerhalb homogener Redundanzgruppen, sondern auf die ganzheitliche Optimalität

der Systemqualität und heterogener Redundanzverfügbarkeit des Gesamtsystems.

Die Exploration wird über die jeweils geltenden Anforderungen der Betriebsmodi aus der Anwendungsdomäne qualitativ parametrisiert. Damit ist eine Betrachtung domänenspezifischer Restriktionen unmittelbar möglich. Die Arbeit zeigt dies exemplarisch für die Automotive-Domäne (Fahrtprofile) sowie ausführlich für ein Raumfahrtsystem (Missionsziele). Durch die Abstraktion technischer Details wird die Rekonfiguration leichtgewichtig auf Ebene architektonischer und qualitativer Auswirkungen in komponentenbasierten Modellen und quantitativer Kennzahlen dokumentiert.

Hierbei nutzt die Arbeit etablierte Methoden zur Exploration des Entwurfsraums entlang einer variantenreichen Initialarchitektur. Die Architektur wird als Palladio-Komponentenmodell entworfen und in dem Architektursimulator Palladio Bench qualitativ bemessen. Innerhalb der Modelle spezifizieren definierte Freiheitsgrade mögliche Variationspunkte. Eine meta-heuristische Optimierung der Konfigurationsoptionen erfolgt in der Palladio-Erweiterung DSE/PerOpteryx unter Einsatz genetischer Algorithmen (NSGA-II), um eine erschöpfende Suche im Entwurfsraum zu vermeiden. Das Vorgehen ermöglicht somit die qualitätsorientierte Erhebung alternativer Konfigurationen im Pareto-effizienten Entwurfsraum. Die Analyse läuft modellbasiert zur Entwurfszeit automatisiert ab.

Folgende Kernbeiträge sind in dieser Arbeit beschrieben:

- Der **Architekturrelationsgraph (***ARG***)** spezifiziert effektive Relationen im Entwurfsraum. Konfigurationen werden dabei als Knoten in absteigender Abhängigkeit zur eingesetzten Sensorik und Aktuatorik sortiert und nach **Rekonfigurationsaufwänden** abgegrenzt. Strukturelle Konfigurationsänderungen geben die Knotenreihenfolge vor. Konkret ausfallgefährdete Ressourcen dienen als Kantenbedingungen, die im Fehlerfall auslösen.
- **Rekonfigurationsentscheidungen** werden entsprechend der Betriebsmodi des Systems parametrisiert. Diese Modi spezifizieren Normalisierungen und Gewichtungen der Qualitätsdimensionen sowie minimaler Akzeptanzwerte. Hieraus werden rein strukturell nicht unterscheidbare Konfigurationsoptionen eindeutig in qualitativen Degradierungen abgegrenzt und ein **deterministischer**

Architekturrelationsgraph (*DARG***)** pro Modus erzeugt. Ein *DARG* bildet dabei den effektiven **Rekonfigurationsraum** des Systems ab.

- Eine **effiziente Exploration im minimierten Raum betriebsoptimaler Alternativen** erfolgt nach Injektion eines Fehlers. Der *DARG* bildet die Basis eines **Expertensystems**, das das Wissen aus dem Entwurf manifestiert und in einem zum produktiven System **synchronisierten Laufzeitmodell** verfügbar macht. Der Ansatz verringert jeweils die Suchkomplexität und fördert zugleich eine transparente Durchführung des Wartungsprozesses bei minimalen Interaktionen mit Entscheidungsträgern. Weiterhin unterstützt der Ansatz dabei die Variation der Betriebsmodi und bietet entsprechende Analyseverfahren an.
- Die praxisrelevante Anwendbarkeit des Ansatzes wird in einer **industriellen Fallstudie** werkzeuggestützt validiert. Dazu wird ein **prototypisches Werkzeug** zur Modellierung, Rekonfigurationserzeugung, Fehlerinjektion und Auswirkungsanalyse auf Basis der theoretischen Grundlage bereitgestellt.
- Weiterhin werden im Rahmen **strukturierter Interviews mit Domänenexperten** reale Entwurfsentscheidungen und betriebliche Wartungsaufwände erhoben und zugleich damit auch die Gültigkeit der Annahmen zur praktischen Anwendung des Ansatzes umfassend beurteilt.

Anhand der exemplarischen Anwendung des Ansatzes zur Rekonfiguration einer adaptiven Regelung der Fahrgeschwindigkeit, wird zunächst die Übertragbarkeit des Ansatzes in die Automotive-Domäne gezeigt. Zur ausführlichen Validierung wird ein software-intensives technisches System aus der Raumfahrtdomäne herangezogen. Als Fallstudie dient das Lageregelungssystem im Kleinsatelliten OOV TET-1 in einer erweiterten Variante mit erhöhter Redundanz und Variabilität in Sensorik und Aktuatorik. Die domänenspezifische Entwurfsraumexploration erfordert die Definition realistischer Modelle, Betriebsmodi, Analysedimensionen und zugehöriger Metriken. Die Validierung untersucht den Einfluss des Ansatzes auf die Aufwandsverringerung in der Kommunikation und den Wartungsaktivitäten des Bodenpersonals. Der Kommunikationsaufwand bezieht sich dabei auf den notwendigen Detaillierungsgrad der übertragenden Datenmengen im Fehlerfall. Der Personalaufwand wird anhand der Komplexität von Entscheidungen zur Rekonfiguration

bemessen. Dabei werden für umfangreiche Fehlersequenzen Kennzahlen im vollständigen und im minimierten Rekonfigurationsraum bestimmt und verglichen. Weiterhin wird eine empirische Studie entlang der Experteninterviews durchgeführt, um die Domänenanforderungen und die Akzeptanz des Ansatzes aufzuzeigen.

Unter Einsatz der prototypischen Implementierung des Ansatzes, wird eine signifikante Reduktion der personellen Aufwände durch die Vermeidung weitreichender Konfigurationsvergleiche von über 90% gegenüber der ungeführten Suche im Wartungsprozess erreicht. Dabei erhebt der Ansatz insbesondere heterogene Redundanzen zwischen Ressourcen und adressiert diese in weitreichenden Rekonfigurationen. Weiterhin werden dabei die Rekonfigurationskosten minimiert, die durch die Reduzierung der Entscheidungsstruktur bei gleichbleibender Gesamtqualität der erhobenen Konfigurationsalternativen entstehen. Die Aufwände für die Kommunikation werden in Bezug auf die Reduktion der übertragenden Daten durch die Verwendung des Laufzeitmodells des Rekonfigurationsraums ebenfalls auf wenige Kilobits pro Wartungsfall reduziert. Die empirische Studie entlang der Experteninterviews zeigt zusätzlich eine generelle Akzeptanz des Ansatzes als Beratungssystem in der Raumfahrtdomäne auf. Die Annahmen für die Domäne werden präzisiert und eine mögliche Einbettung des architekturzentrierten Konfigurationsbegriffs in den Wartungsprozess wird erfolgreich belegt. Dabei werden die kontinuierliche Repräsentation des Systemzustands und die transparente Begründung von Rekonfigurationsvorschlägen für Entscheidungsträger positiv hervorgehoben.

Abstract

In many application domains of software-intensive technical systems, complex devices and software modules must fulfil strong requirements. In particular, sensor and actuator hardware resources need to be integrated in redundant constellations with homogeneous devices or heterogeneous replacements. Such systems are equipped with hardware redundancies to guarantee systems' resilience with proper sensing, control and actuation even in hostile environments. In automotive systems, continuous operation during autonomous driving is crucial to prevent any human casualties.

Besides functionality, high quality of software subsystems is an important issue to meet strong quality requirements. In particular, for long-term objectives, the assurance of reliability is crucial for success. However, in case of system defects, self-maintenance is only feasible to a limited extent. To support such maintenance activities, techniques for detection, isolation and recovery of faults are state of the art in today's software-intensive technical systems. Usually, the software features emergency operation modes to guarantee at least communication abilities.

For reconfiguration beyond homogeneously redundant resources, manual decision-making is necessary, resulting in down-times, communication effort and man-hours in maintenance phases, because the fault recovery is primarily comprises configuration adjustments by human personnel. This may lead to increasing delays and makes broad maintenance interfaces inevitable. In the worst case, a lengthy interruption or insufficient capabilities of remote maintenance results in a total loss of the system. In addition, many asset combinations (system configurations) with contradicting quality objectives need to be considered to enhance the space of possible design options. Each feasible configuration should be investigated, easily leading to an exponential design space. This is even more serious by anticipating hardware faults in fault-tolerant systems. Each fault might results in expensive changes in

deployments of stressed resources in redundant constellations. Identification and evaluation of the best-fitting configuration, neither with large reconfiguration effort nor quality loss, remain computationally intensive and difficult tasks. Hence, concepts for autonomic fault handling are necessary to increase self-adaptive maintenance of such expensively maintainable systems.

This thesis proposes an approach for handling faults of sensing and actuating resources by self-reconfiguration on the level of functional architecture at runtime supporting quality degradation. Thus, the approach addresses automated reconfiguration decision support determining a variety of Pareto-efficient architectures according to variable hardware availability and quality properties. Reconfiguration options for control software, according to available sensing and actuation resources, are derived and prioritised with respect to predicted qualitative impacts and reconfiguration efforts. The approach provides a formal structure to describe multiple configurations of a system continuously adapting to changing environmental circumstances. A configuration includes the structure of functional components – as part of a component-based architecture – and their relations to corresponding hardware resources. Thus, the knowledge about relations of the system's variations is persisted in a decision model at design-time on the level of software architectures. These architectures may differ considerably due to the usage of different subsets of hardware resources.

The approach distinguishes between configurations on a higher level of abstraction by means of distances between qualitative ratings and configuration changes for architectures. The abstraction relies upon a combination of differing hierarchical cluster analyses and design space exploration with genetic algorithms. The analyses yield a hierarchical state-based graph structure, the Deterministic **Architectural Relation Graph (***DARG***)**, defining possible reconfiguration paths for supporting decisions at runtime. Nodes represent quality-rated configurations and edges relate configurations among one another. A transition from one node to another is triggered by hardware resource faults. The computation of the ratings for Pareto-efficient configurations is done at design-time to reduce evaluation efforts during operation. Upon a resources fault, the model is traversed for an alternative architecture. Thus, if a hardware resource of the system is no longer available at runtime, a suitable alternative configuration is determined at low expenses. This constitutes an **Expert System** for a transparent analysis of available deployments as well as an acceleration of the reconfiguration process during the

maintenance phase. The activities of triggering, determining, and deploying new reconfigurations are implemented as a control loop.

The architecture is designed as a Palladio component model and evaluated by the architectural simulator Palladio Bench. Established methods are used for exploring the design space of a variant-rich initial architecture. Several degrees of freedom define possible points for variations within the model. The Palladio extension DSE/PerOpteryx performs a meta-heuristic optimisation to explore possible configuration options in the design space with Genetic Algorithms (NSGA-II) while avoiding an exhaustive search. Thus, the procedure enables a quality-oriented examination of alternative configurations in the Pareto-efficient design space. Based on the theoretical framework, I provide tool support for analysis of related architectures and a concept for reconfigurations during operation.

For the evaluation, the approach is applied to a subsystem of a spacecraft from industry. The attitude control system of the OOV TET-1 micro satellite bus is examined in a reengineered variation with heterogeneous redundancies for sensing and actuation. In this way, the capability to extend the autonomy of the satellite is analysed within the approach. In addition, a comprehensive empirical study is performed to identify domain restrictions for the general acceptance of the approach.

Danksagungen

Viele Personen haben mich in meiner Zeit als wissenschaftlicher Mitarbeiter an der TU Braunschweig und als externen Doktorand am KIT begleitet.

Meinem Doktorvater Herrn Prof. Dr. Ralf Reussner möchte für die langjährige Unterstützung in der Promotion und der guten Zusammenarbeit im „DFG Schwerpunktprogramm 1593“ danken. Dabei bin ich Ralf vor allem für die freiwillige Übernahme meiner Betreuung und die unkomplizierte Integration in seine Arbeitsgruppe sehr dankbar. Ausdrücklich möchte ich auch Frau Jun.-Prof. Dr. Anne Koziolek für ihr Engagement in meiner wissenschaftlichen Betreuung in vielen Einzelgesprächen und Publikationen danken. Ich habe in den Jahren am Lehrstuhl „Software Design and Quality“ (SDQ) sehr viel gelernt und konnte mein Promotionsthema umfassend schärfen. In diesem Kontext möchte ich auch allen Kollegen am SDQ und dem „Forschungszentrum für Informatik“ (FZI) für die stets freundliche Aufnahme und die vielen fachlichen Ratschläge danken. Ich habe mich immer sehr wohl bei euch gefühlt.

Mein herzliches Dankeschön richtet sich an Frau Prof. em. Dr. Ursula Goltz als ehemalige Leiterin des „Instituts für Programmierung und Reaktive Systeme“ (IPS) und Initiatorin meiner Promotion. Ohne Ulla hätte ich die Laufbahn als wissenschaftlicher Mitarbeiter vermutlich nicht eingeschlagen.

Für die kommissarische Fortsetzung des Institutsbetriebs des IPS und der damit verbundenen Sicherung meiner Stelle als wissenschaftlicher Mitarbeiter, bedanke ich mich bei Herrn Prof. Dr. Wolf-Tilo Balke. Ihm und Frau Prof. Dr. Ina Schaefer möchte ich für die zuvorkommende und unkomplizierte Beratung in persönlichen und inhaltlichen Belangen danken. Ina sei zusätzlich zur Übernahme des Koreferats dieser Arbeit herzlich gedankt. Während meiner Zeit an der TU Braunschweig, durfte ich in einem fairen, angenehmen und kollegialen Umfeld arbeiten. Meinen langjährigen Kollegen am IPS gebührt hier ein besonderer Dank. Dabei möchte ich Herrn Dr. Malte Lochau

danken, der mich im Rahmen meiner Diplomarbeit und in meinen Anfängen als Promotionsstudent wegweisend beraten und unterstützt hat. Auch Herrn Dr. Matthias Hagner gilt mein ausdrücklicher Dank für seine motivierende Beratung und Zusammenarbeit in Forschung und Lehre. Weiterhin danke ich Herrn Stephan Mennicke für die vielen fruchtbaren Diskussionen zu Forschungs- und Lehrthemen sowie für sein stets offenes Ohr. Herrn Dr. Hauke Baller, Herrn Sascha Lity und Herrn Bernhard Witte gilt mein Dank für die vielen kollegialen Gespräche sowie die Kooperationen in Forschung, Lehre und organisatorischen Themen. Herrn Nils-André Forjahn danke ich für seine langjährige Unterstützung in der Erweiterung eines prototypischen Werkzeugs im Rahmen meines Promotionsthemas.

Im Kontext der anonymisierten Interviews danke ich den befragten Domänenexperten der „Astro- und Feinwerktechnik Adlershof GmbH“ für ihre Beteiligung und die mir entgegengebrachte Diskussionsbereitschaft.

Ein besonderer Dank geht an meine Eltern, die mich bis heute in vielen Lebensphasen intensiv unterstützten. Hervorheben möchte ich dabei die Ermöglichung meines Studiums in Braunschweig und die Motivation zur akademischen Laufbahn. Abschließend richtet sich mein liebevoller Dank an meine Ehefrau und meine Kinder. Ohne den emotionalen Rückhalt, die Rücksicht und die Liebe, hätte ich meine Promotion nicht absolvieren können.

All diese Personen haben ihren speziellen Teil zum Abschluss meiner Promotion beigetragen. Meinen herzlichen Dank dafür.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Diese Arbeit beschreibt einen neuartigen Ansatz zur Wartungsunterstützung software-intensiver technischer Systeme. In diesem Rahmen werden nachfolgend Zielsetzungen, Annahmen und mögliche Anwendungsdomänen vorgestellt. Abschließend wird ein durchgängiges Beispiel eingeführt.

1.1. Motivation

Software übernimmt im zunehmenden Maße die Steuerung, Regelung und Überwachung heutiger sicherheitskritischer Infrastrukturen, Geräte, Anlagen und Prozesse. Ausfälle innerhalb solcher *software-intensiven technischen Systeme*, vergleiche Abschnitt 2.1.1, führen zu erheblichen wirtschaftlichen Schäden und können schlimmstenfalls Menschen und Umwelt gefährden. Daher ist im Entwurf software-intensiver technischer Systeme die inhärente Fehlertoleranz von wesentlicher Bedeutung.

Bedingt durch schwankende äußerliche Einflüsse und zeitkritischen Anforderungen, wird im Systementwurf auf zuverlässige Software und Hardware zurückgegriffen. So werden zur Ausführung einer intensiv getesteten Betriebssoftware vorrangig Hardwareressourcen eingesetzt, die in dem erwarteten Umfeld bereits erprobt sind, vergleiche Abschnitt 7.1. Zur weiteren Kompensation von Teilausfällen werden Ressourcen in redundanten Konstellationen integriert, vergleiche [Avi97]. Dies ermöglicht eine Fortsetzung des Betriebs bis zu einem gewissen Grad selbst dann noch, wenn kritische Bauteile ausfallen. Die Stabilität der Funktionalität und der Systemqualität steht im Vordergrund. Bei der Kompensation eines Totalausfalls sind jedoch Degradierungen beider Aspekte möglich.

Auf Basis eines fehlertoleranten Systementwurfs werden umfangreiche Analysen über Wahrscheinlichkeiten und Folgen von Ausfällen einzelner Subsysteme durchgeführt. Dabei spielen Erkenntnisse aus Datenblättern und Vorgängersystemen eine wesentliche Rolle, vergleiche Abschnitt 7.1. Die Auswahl und Redundanz der Ressourcen leitetet sich aus den Kundenanforderungen des Funktionsumfangs und der Zuverlässigkeit sowie den technischen Rahmenbedingungen zur Ausführung ab.

Im Betrieb fehlertoleranter Systems treten domänenbedingt Fehlerfälle ein. Im Allgemeinen sind Ausfälle in redundanten homogener Baugruppen, vergleiche Abschnitt 2.1, durch vorgesehene Strategien zur Fehlerbehandlung mit geringer Ausfallzeit zu beheben. Fallen komplette Baugruppen aus, besteht in der Wartung jedoch ein Entscheidungsproblem in der effizienten Auswahl einer adäquaten Konfiguration mit abweichendem Ressourcenbedarf. Eine Alternative muss einerseits die Betriebsanforderungen effektiv erfüllen und anderseits auf der aktuell verfügbaren Ressourcenplattform ausführbar sein. Die Effizienz richtet sich dabei nach dem Aufwand zur Identifikation der Konfigurationen und dem Übergang in diese. Eine effektive Alternativkonfiguration nutzt die Ressourcenplattform auf eine Weise, die zu einer verbesserten qualitativen Erfüllung der geltenden Betriebsanforderungen führt.

Am Beispiel des prominenten *Kepler Weltraumteleskops* kam es zu einer aufwendigen Anpassung der Systemkonfiguration in der Wartung. Nach dem Verlust der Hälfte der Hardwareressourcen zur künstlichen Drehmomenterzeugung, war eine Lageregelung des Teleskops unmöglich. Die Mission galt als gescheitert, jedoch wurde nach einer möglichen alternativen Konfiguration weiterhin gesucht. Durch eine aufwendige Anpassung der Software [HSH+14] und deutliche Lockerung der Missionsanforderungen, konnte schließlich ein alternativer Betriebsmodus definiert werden. Die Abbildung 1.1 zeigt das Szenario „Kepler's Second Light". Unter Verwendung der zwei verbleibenden Aktuatoren (im Bild: Reaction Wheel 1 und 3) und Berücksichtigung des Photonendrucks im Sonnenlicht ist es möglich, das Teleskop in regelmäßigen Abständen stabil auszurichten. Ein Ingenieur der NASA vergleicht das aufwendige Verfahren mit dem Balancieren eines Stifts auf der Fingerspitze. Eine Fortsetzung der Mission ist zwar möglich, jedoch mit verringerter Observationsleistung. Anstelle des Totalausfalls, kommt eine Degradierungsstufe zum Tragen.

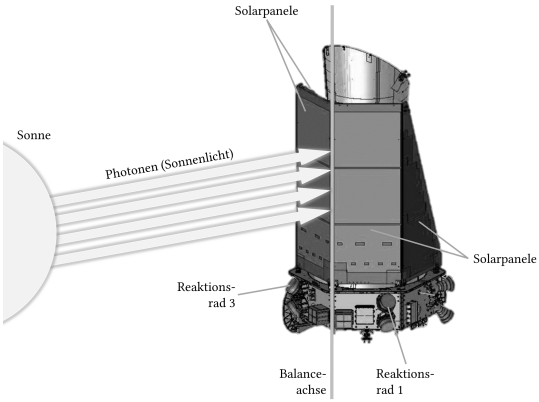


Abbildung 1.1.: Rekonfiguration des Kepler Teleskops (Quelle: NASA Ames / W. Stenzel, `www.nasa.gov`, angepasst)

Die Exploration des Raums möglicher Alternativen bedarf der Validierung sämtlicher Konfigurationen und deren Abwägung der Adäquatheit zur qualitativen Zielerreichung und Einhaltung von Budgets für strukturelle Änderungen auf Ebene der Ressourcenplattform. Dies führt im Allgemeinen zur hohen personellen Kosten. Ist der Systemzugriff zusätzlich durch Distanzen beschränkt, treten zusätzliche Aufwände in der Kommunikation in Form von Latenzen und einem beschränkten Zugriff über vordefinierte Diagnoseschnittstellen auf.

Aktuelle Ansätze zur Fehleranalyse untersuchen bereits frühzeitig im Entwurf kritische Systemkonfigurationen. Dabei werden Rekonfigurationsregeln vorrangig innerhalb fest definierter homogener Redundanzen festgelegt. Die Erstellung solcher Prognosen führt bereits zu signifikanten Aufwänden in der Entwicklungsphase. Im Kontext langlebiger Systemen, insbesondere bei spezialisierten Individualentwicklungen, ist eine Intensivierung der zusätzlichen Vorausberechnung weitreichender Rekonfigurationsentscheidungen trotz erhöhter Entwurfsaufwände jedoch aus operativer Sicht lukrativ. So kompensieren sich mit steigender Laufzeit die initialen Investitionen zunehmend.

Das Wissen aus den Fehleranalysen zur Laufzeit wird vorrangig zur Festlegung von homogenen Redundanzen genutzt und der Beurteilung von Ausfallraten. Eine Integration und Nutzung der Daten in einer Wissensbasis zur kosten- und qualitätsoptimalen Rekonfigurationen für heterogene Redundanzen in der Wartung erfolgt nicht. Ein Ansatz der zugleich einen aufwandsoptimierten Konfigurationsraum entsprechend der noch verfügbaren Sensorik und Aktuatorik darstellt und zudem nach qualitativen Eigenschaften ordnet, kann die Wartung eingebetteter Systemen unterstützen. Hierbei ist es notwendig, dass das Analysewissen aus der Entwurfszeit für den Betrieb des Systems und dessen Varianten analysiert, geordnet und persistiert wird. Zeitgleich kann ein Ansatz auf der Architekturebene die Transparenz des Rekonfigurationsprozesses durch das nachvollziehbare Aufbereiten von Änderungsschritten fördern.

1.2. Forschungsgegenstand

Diese Arbeit untersucht existierende Wartungsprozesse software-intensiver technischer Systeme und beschreibt einen Ansatz zur Verlagerung von Aufwänden zur Laufzeit in die Entwurfszeit. Dabei werden voranging kosten- und qualitätsorientierte Unterstützungskonzepte für personelle Wartungsaktivitäten und Kommunikationsaspekte untersucht.

1.2.1. Herausforderungen

Software-intensive technische Systeme sind beispielsweise im Weltall extremen äußerlichen Einflüssen oder in Produktionsanlagen intensiver Abnutzung ausgesetzt. Sie verrichten jedoch zeitgleich hochsensible und zeitkritische Arbeiten. Dies hat Hardwareausfälle zur Folge und stellt hohe Anforderungen an die Wartung solcher Systeme. Die zumeist kostspieligen manuellen Wartungsaktivitäten sind jedoch durch Zugänglichkeit, Diagnoseschnittstellen und Kommunikationseinschränkungen limitiert. Bis der Fehler manuell kompensiert wurde, ist im Allgemeinen eine Fortsetzung von wissenschaftlichen Experimente oder Produktionszyklen nicht möglich. Folglich kommt es bei größeren Ausfällen zu einem signifikanten Verlust von Missions- oder Produktionszeit.

1.2.2. Lösungsansatz

Ein Ansatzpunkt zur Verbesserung der Wartung ist die frühzeitige Berücksichtigung von umfassenden Qualitätssicherungsmöglichkeiten in Hardware und Software über die Entwurfszeit hinaus. Die Manifestation und Aufbereitung des Entwurfswissens für die Wartung folgt der Idee einer *Knowledge Carrying Software* vergleiche [GRG+14]. Die Entwurfsmodelle werden dazu um zusätzliche Kontextinformationen zu Redundanzbeziehungen, Rekonfigurationskosten und qualitativen Degradationen angereichert. Insbesondere ist dabei eine Fähigkeit zur selbstständigen Erhebung und Begründung von Konfigurationsalternativen im Fehlerfall vorzusehen, vergleiche Selbstmanagement in Abschnitt 2.2.5.

Ausgehend von einem hohen Grad an Hardwareredundanz, setzt das System funktional möglichst gleichbleibend den Betrieb fort, qualitativ (zum Beispiel in Bezug auf Antwortzeiten) wären Abstriche jedoch akzeptabel. Dies würde manuelle Wartungsaktivitäten weitestgehend vermeiden und Ausfallzeiten reduzieren. Die Abbildung 1.2 veranschaulicht die Kernaspekte des Lösungsansatzes. Der Prozess besteht aus der Modellierung des Systemmodells, einer Analyse auf den Modellen zur Bildung des Rekonfigurationsraums und der Laufzeitexploration zur Identifikation von Konfigurationsalternativen. Alle notwendigen Berechnungen in der Analyse finden als Prognosen zur Entwurfszeit statt.

Das System wird mittels Techniken des komponentenbasierten Software Engineering in einer Softwarearchitektur erfasst. Dabei werden Beziehungen zwischen Softwarekomponenten und einer Ressourcenplattform hergestellt. Die Plattform gibt Restriktionen innerhalb Redundanzgruppierungen vor und stellt technische Spezifikationen zu jeder Ressource bereit. Unter Einsatz eines suchbasierten Sampling-Verfahrens werden syntaktisch valide Ausprägungen der Softwarearchitektur in einer multiteriellen Optimierung selektiert. Die qualitativen Eigenschaften werden aus den technischen Spezifikationen der Ressourcen extrahiert. Eine Sortierung der Architekturausprägungen nach deren Pareto-Effizienz, vergleiche Abschnitt 2.2.3, erfolgt weiterhin in diesem Schritt. Abschließend werden ganzheitliche Nutzwerte für jede Architektur mittels eines Ansatzes zur Zielkonfliktresolution für ein definiertes Benutzungsszenario berechnet.

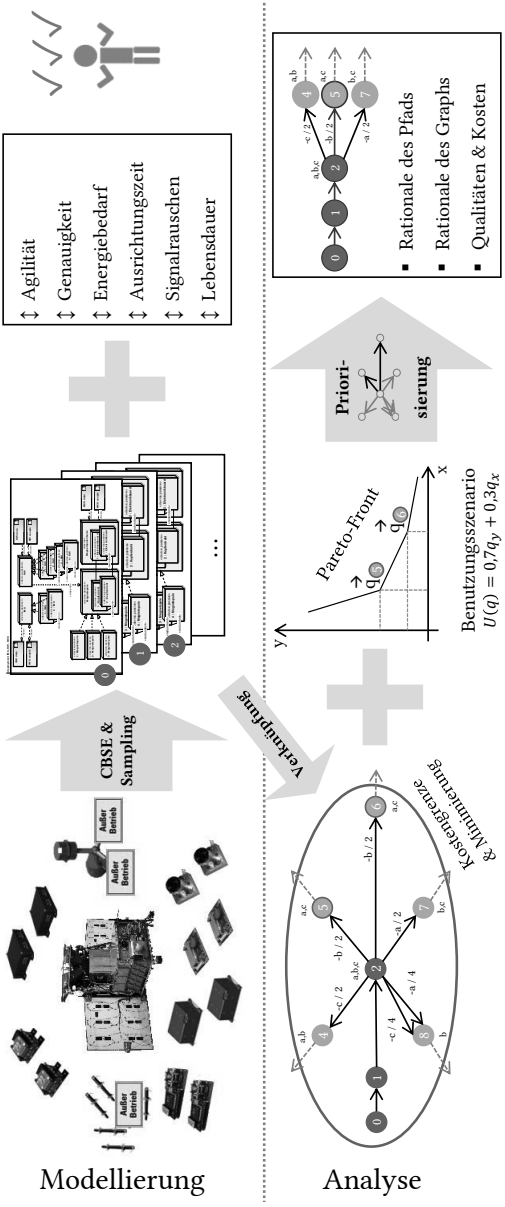


Abbildung 1.2.: Übersicht Lösungsansatz

Die Analyse greift die bewerteten Architekturausprägungen auf und analysiert deren Gemeinsamkeiten auf Basis der Schnittmengen in der Ressourcenplattform. Dabei werden Architekturen ihrer strukturellen Distanz nach in einem Graphen in Relation gesetzt. Durch die Vorgabe einer Kostengrenze für Ressourcenänderungen innerhalb einer Rekonfiguration wird der Rekonfigurationsraum frühzeitig beschränkt. Weiterhin werden redundante Kanten eliminiert und transiente Architekturen für mehrschrittige Rekonfigurationen genutzt. In dem minimierten Graph wird unter Berücksichtigung der Nutzwertdifferenzen ein Determinismus zur Wahl von Konfigurationsalternativen erzeugt. Dabei werden insbesondere Pareto-effiziente Architekturen in der Auswahl bevorzugt.

Tritt im Betrieb ein Fehler auf (hier: Ressource „b“) wird in dem betreffenden Subgraphen eine alternative Konfiguration erhoben, die zugleich kostengünstig erreichbar ist und den betriebsoptimalen Nutzwert (hier: $U(5) > U(6)$) bietet. Abschließend resultiert ein Subgraph mit den nächstmöglich erreichbaren Alternativen und einer vollständigen Historie über erfolgte Rekonfigurationen. Zusätzlich werden diverse Kennzahlen zur Begründung der Entscheidung bereitgestellt, die sich auf den Verlauf der Exploration und den aktuellen Zustand des Rekonfigurationsraums beziehen.

1.2.3. Forschungsfragen

Die Vorteile des neuartigen Ansatzes liegen in der effizienten Lösungsfindung bei geringerem Kommunikationsaufwand bezüglich Rekonfigurationsinformationen, kurzen Ausfallzeiten und reduziertem Personaleinsatz. Durch die architekturorientierte Ausrichtung des Ansatzes lässt sich zudem die Nachverfolgbarkeit und Verständlichkeit durch eine abstrakte Modellierung von Regeln zur Fehlerbehandlung steigern. Hierbei lässt sich die Ermittlung einer alternativen Systemkonfiguration beispielsweise durch Rekonfigurationspfade (Englisch: Traces) in einem Entscheidungsgraphen darstellen.

FF1 *„Welche Faktoren limitieren die Voraussagekraft der Konfigurierbarkeit eines Systems unter Annahme kontextfreier Fehlerszenarien, festgelegter Betriebsmodi und statischer Qualitätseigenschaften?“*

FF2 *„Lassen sich Fernwartungsaufwände im Betrieb durch approximative Priorisierungen von Rekonfigurationsentscheidungen zur Entwurfszeit signifikant reduzieren?“*

FF3 *„Lässt sich die Prozesstransparenz in der Fernwartung durch den Einsatz komponentenbasierter Modelle effektiv unterstützen?“*

FF4 *„Liefern meta-heuristische Suchverfahren auf Architekturbasis eine adäquate Entscheidungsbasis zur Festlegung weitreichender, nicht trivial identifizierbarer, Systemkonfigurationen mit qualitativen Abstufungen?“*

1.2.4. Zielsetzungen

Der Ansatz ermöglicht die autonome Identifikation adäquater Konfigurationen anstelle von kostenintensiver manueller Auswahl durch Fachexperten. Der hohe Abstraktionsgrad wird durch eine transparente Sicht auf den Auswahlprozess ermöglicht. Hierzu werden Systemkonfigurationen auf Ebene von Softwarearchitekturen in Verbindung mit Anforderungen an zur Ausführung benötigten Hardwareressourcen dargestellt. Um einen effizienten Wechsel zwischen Konfigurationen zu gewährleisten, werden einzelne Konfigurationen in einem Entscheidungsgraphen miteinander verknüpft. Die Übergänge werden dabei so gewählt, dass eine Entscheidungsfindung effizient stattfinden kann, das heißt die Entscheidungsstruktur ist nicht vollvermascht und auf relevante Kanten minimiert.

1.3. Einordnung und Annahmen

Die Einordnung des Ansatzes und die Annahmen zu dessen Anwendung werden nachfolgend erläutert.

1.3.1. Einordnung

Zur Einordnung des Ansatzes wird die Relevanz des Problems beschrieben und einer Beurteilung der Adäquatheit und Stärke der Modellbildung durchgeführt.

Relevanz des Problems: Durch räumliche Distanz, baulich bedingten Zugangsbeschränkungen in der Diagnose oder hohe Losgrößen der Systeme, gewinnt die Fernwartung an Bedeutung. So wird der direkte Systemzugriff zum Austausch defekter Bauteile durch räumliche Distanzen, bauliche Gegebenheiten oder der Unwirtschaftlichkeit von Rückrufaktionen beschrankt. Diese Restriktionen treten beispielsweise in der Raumfahrtdomäne, der Automatisierungstechnik für Produktionsanlagen beziehungsweise der Automotive-Domäne auf. Insbesondere bei fehlertoleranten Systemen, die aufgrund störungsintensiver Betriebsbedingungen auf eine Vielzahl von Ressourcenausfällen reagieren müssen, ist eine inhärente Rekonfigurierbarkeit zur Steigerung der Verfügbarkeit vorzusehen. Zur Beurteilung adäquater Entwurfsräume sind umfangreiche Analysen zur Alternativenauswahl notwendig. Technologische und rechtliche Grenzen bedingen eine frühzeitige Bestimmung von Größe und Flexibilität des Raums möglicher Rekonfigurationen. Zugleich muss jederzeit eine transparente Darstellung für die Entscheidungsträger möglich sein. Aus diesem Kontext entstehen hohe Anforderungen einerseits an die Zuverlässigkeit der Rekonfigurationen und andererseits an die Dokumentation und Effektivität der Automatismen. Zur geplanten Rekonfiguration fehlertoleranter Systeme ist eine Beurteilung adäquater Lösungskandidaten in einem komplexen Entwurfsraum notwendig. Aus Komplexitätsgründen ist dieses Problem weder erschöpfend zu berechnen, noch zur Laufzeit des Systems effizient lösbar. Der entwickelte Ansatz adressiert die Approximation der notwendigen Berechnungen auf Ebene von Architekturmodellen, um analytische Wartungstätigkeiten von der Laufzeit in die Entwurfszeit zu verlagern. Durch diese Vorausberechnung lassen sich Entscheidungen in der Wartung beschleunigen und strukturbezogen argumentieren.

Adäquatheit des Modells: Der Ansatz basiert auf Modellen, die vorrangig die Struktur des Systems auf Ebene von Komponentenmodellen, abstrakten

Ressourcen und deren Bindungen zueinander beschreiben. Dieser Abstraktionsgrad ermöglicht eine hinreichende Beschreibung von Gemeinsamkeiten und Abweichungen zwischen Systemkonfigurationen und zugleich eine transparente Sicht auf das System. Im Ansatz stehen die Beziehungen zwischen Basiskomponenten der Software und den Ressourcen der Ressourcenplattform zur Beurteilung der Ausführbarkeit einer Konfiguration im Vordergrund. Weiterhin werden Konfigurationen neben den Ressourcenanforderungen auf struktureller Ebene voneinander abgegrenzt. Diese Vergleiche untersuchen ausschließlich Änderungen in den Assemblierungskontexten von Basiskomponenten. Eine ausführliche Spezifikation von Kontroll- und Datenfluss innerhalb der Basiskomponenten ist nicht notwendig. Die zusätzlichen Aufwände zur Definition der Modelle im Entwurf führen zu reduzierten Aufwänden in der Wartung. In der produktiven Anwendung existiert zu jedem abstrakten Modell eine reale Implementierung. Diese Verknüpfung stellt der Anwender manuell her. Dabei ist es für den Ansatz grundsätzlich irrelevant, ob die Implementierung frühzeitig im Entwurf verfügbar ist oder erst im Betrieb nachträglich entwickelt wird. Sollten allerdings Messungen innerhalb einer prototypischen Implementierung zur Verfeinerung von Prognosedaten wieder in den Entwurf einfließen, entstehen iterative Abhängigkeiten.

Stärke des Modells: Auf Grundlage des variantenreichen Entwurfs ist eine Menge von untereinander zunächst nicht im Zusammenhang stehenden Konfigurationsalternativen verfügbar. Durch die Anwendung des Ansatzes werden Gemeinsamkeiten zwischen den Alternativen identifiziert und in einer Graphstruktur explizit festgehalten. Dabei werden einerseits Rekonfigurationskosten minimiert und zugleich qualitative Aspekte beachtet. Das Ergebnismodell liefert, entgegen dem originären Suchraum, konkrete Gegenmaßnahmen für induzierte Ressourcenfehler. In Abhängigkeit eines Fehlers wird so, neben einer adäquaten Alternativkonfiguration, zugleich auch der Pfad notwendiger Rekonfigurationen aufgezeigt. Die Stärke der Entscheidungsstruktur liegt dabei in der Abwägung von ausschließlich anwendbaren Alternativen, dem Nominieren genau einer Lösung, der Darstellung von resultierenden qualitativen Degradationen sowie der Dokumentation erfolgter Rekonfigurationen.

Nützlichkeit: Der Anwender wird im Wartungsprozess unter Anwendung des Ansatzes entlastet, hingegen steigen aber die Tätigkeiten in der Entwurfsphase. Im Kontext des generellen massiven Anstiegs der Wartungskosten, im Vergleich zu den Kosten der initialen Entwicklung nach Zelkowitz et al. [ZSG79], lässt sich auch die Nützlichkeit dieses Ansatzes beurteilen. Insbesondere im Kontext der Zieldomänen im Bereich langlebiger Software-/Hardware, ist ein Mehraufwand im Entwurf als akzeptabel zu bewerten. Weiterhin lässt sich der Ansatz auch zur Beurteilung der Effektivität möglicher Ressourcenredundanzen nutzen. Somit ist der Mehraufwand nicht ausschließlich zusätzlich aufzubringen, sondern deckt auch einen Teil der notwendigen Analysen im Entwurf fehlertoleranter Systeme ab.

1.3.2. Kontextannahmen

Nachfolgend werden die Annahmen zur Anwendung des Ansatzes detailliert eingeführt.

Architekturorientierter Entwurf: Das betrachtete fehlertolerante System ist hinreichend mit einem Architekturentwurf dokumentiert. Eine Ableitung einer Architektur aus anderen Entwurfsdokumenten ist alternativ auch hinreichend. Ein komponentenorientierter Entwurf sollte zumindest grundlegend eingesetzt werden. In jedem Fall müssen Konfigurationen auf Architekturebene abgrenzbar und eindeutig identifizierbar sein. Somit werden Rekonfigurationen zwischen Architekturen beschreibbar.

Vordefinierte Betriebsszenarien: Alle Betriebsszenarien des betrachteten Systems müssen zur Entwurfszeit bekannt sein. Der spätere Einsatzzweck und die erwartete Umgebung muss bekannt sein, um qualitative Mindestanforderungen abzuleiten und zur Nutzwertbildung heranzuziehen.

Sensorik und Aktuatorik spezifiziert: Als Ressourcen betrachtet der Ansatz ausschließlich Sensorik und Aktuatorik. Einerseits sind in diesem Zusammenhang komplexe Redundanzbeziehungen beobachtbar, andererseits treten diese Ressourcen in großer Stückzahl in fehlertoleranten Systemen auf. Folglich liegt ein großer Raum an Variationen vor, der den Einsatz des Ansatzes

zielführend argumentiert. Der Ansatz erfordert, dass für jede Ressource ausführliche technische Informationen vorliegen, um eine Ableitung von qualitativen Eigenschaften zu ermöglichen. Die Daten können dabei sowohl aus Datenblättern, als auch aus prototypischen Simulationen oder Erfahrungen aus Vorgängersystemen stammen.

Zuordnung von Sensor-/Aktuatorkomponenten eindeutig: Es ist bekannt, welche Ressourcen als Datenquellen und Datensenken für Softwarekomponenten benötigt werden. Konkret müssen die Hardwareressourcen auf eindeutige Repräsentationen innerhalb der Softwarearchitektur auszubilden sein. Dies ist erforderlich um den Austausch oder Wegfall einzelner Ressourcen auf Architekturebene auszudrücken.

Diskretes Fehlermodell und erwartete Fehlersequenz: In dieser Arbeit werden Fehler, in Form ausfallender Ressourcen, als persistent angenommen. Die Menge der Ressourcen in der Ressourcenplattform fällt ab und steigt niemals im Betrieb des Systems wieder an. Dabei wird grundsätzlich von einer erwarteten Eintrittswahrscheinlichkeit bekannter Fehler ausgegangen. In späteren Analysen wird von der Annahme einer Fehlersequenz teilweise abgewichen. Durch eine partielle Permutation werden alle möglichen Ressourcen als fehlerhaft eingestuft.

Ausschließlich gültige Konfigurationen in Freiheitsgraden: Es werden nur gültige Konfigurationen durch Freiheitsgrade beschrieben. Das Softwaremodell wird dabei stets als korrekt angenommen und die Ressourcen zur Entwurfszeit als völlständig verfügbar. Die Restriktionen innerhalb der Ressourcenplattform beschränken die Freiheitsgrade, dies wird jedoch in der Analyse berücksichtigt.

1.4. Auswahl möglicher Anwendungsdomänen

Das Anwendungsgebiet des Ansatzes liegt in der Wartungsunterstützung eingebetteter Systeme mit hochgradig vernetzten redundanten Sensoren

und Aktuatoren. Mögliche Anwendungsdomänen werden nachfolgend im Kontext des Ansatzes betrachtet.

1.4.1. Automotive Fahrerassistenzsysteme

Mit zunehmender Softwareintensivierung im Fahrzeug, steigt auch der Vernetzungsgrad mit der Umwelt. So nutzen moderne Fahrzeuge eine ständige Internetverbindung, um Komfortdienste, Notruflösungen und kontinuierliche Software-Updates anzubieten. Wo sich früher der Fahrzeugzugriff auf eine Hauptuntersuchung[1] alle zwei Jahre konzentrierte, sind heutzutage Änderungen am Softwaresystem jederzeit aus der Ferne möglich. Wartungen an Hardware oder Mechanik erfordern weiterhin den direkten Zugriff im Rahmen von Inspektionen oder kostenintensiven Rückrufaktionen.

Trotz hoher Stückzahlen, starken Individualisierungen und unterschiedlichsten Umweltbedingungen, ist dem Kunden jederzeit ein uneingeschränktes Fahrerlebnis und zuverlässige Betriebssicherheit zu bieten. Die Betriebssoftware in der Automotive-Domäne muss sich daher adaptiv an veränderliche Umweltbedingungen anpassen und sicherheitskritische Aufgaben ausführen. Eine qualitative Degradation, zum Beispiel Notbetriebsmodus mit verringerter Fahrgeschwindigkeit, ist möglichst zu vermeiden. Dieses Szenario verschärft sich bei voll- oder teilautonomen Funktionalitäten. Fahrerassistenzsysteme (Englisch: Advanced Driver Assistance Systems, ADAS) realisieren eine selbstständige Anpassung des Fahrbetriebs in Interaktion mit dem Fahrer. In dieser Disziplin besteht eine wesentliche Herausforderung in der Erfassung und Interpretation des Fahrzeugumfelds. Hierzu werden eine Reihe unterschiedlicher Sensoren eingesetzt. Neben physikalischen Redundanzen, vergleiche Abschnitt 2.1, nutzten Ansätze der Sensorfunktion den Überschnitt von Sensordaten um aus heterogenen Kombinationen notwendige Informationen abzuleiten, vergleiche [Ohl14].

Der in dieser Arbeit verfolgte Ansatz könnte die Identifikation und Abwägung von Konfigurationen mit variierenden Sensoreinsatz unterstützen und qualitativ optimieren. Neben der Identifikation und Erläuterung von Entwurfsentscheidungen wäre eine autonome Umsetzung der Entscheidungen weiterhin denkbar.

[1] Hier sei angenommen, dass zugehörige Inspektionen beim Werkstätten des Herstellers durchgeführt werden.

Das Laufbeispiel dieser Arbeit, vergleiche Abschnitt 1.5, stammt aus der genannten Automotive-Domäne.

1.4.2. Produktionssysteme

Im Umfeld der Automatisierung von Produktionsprozessen werden technische Anlagen derart betrieben, dass maximale Durchsatzraten mit minimalem Ausschuss entstehen. Zur Steuerung des Prozesses wird ein umfangreiches Netzwerk aus Sensoren, Aktuatoren und Steuereinheiten aufgebaut. Die Entwicklung Richtung „Industrie 4.0“ intensiviert die Vernetzung der Anlagen untereinander und zur Peripherie in „Smart Factories“. So werden Produktionsprozesse adaptiv gesteuert, bis hin zur Disposition von zeitgerechten Lieferungen.

Ein Totalausfall einer Anlage in dieser Abhängigkeitskette hat massive finanzielle Folgen und ist daher zu vermeiden. Der unterbrechungsfreie Betrieb führt zu einer enormen Belastung für einzelne Bauteile und erschwert Wartungstätigkeiten. Der Austausch von Bauteilen erfordert, dass die Produktion währenddessen weiterläuft. Dies kann beispielsweise durch eine Verlagerung der Materialflüsse auf redundante Mechanik und Elektronik erfolgen. In diesem Fall sind qualitative Degradationen anstelle eines kompletten Produktionsausfalls denkbar.

Ohne konkrete Betrachtung der mechanischen Aspekte, unterstützt der Ansatz die Voraussage von qualitativen Degradationen bei variierenden Ressourceneinsatz. Konkret kann der Ansatz bereits in der Entwurfsphase zur Erhebung möglicher Entwurfsalternativen zur Wartungsplanung genutzt werden.

1.4.3. Raumfahrtsysteme

Raumflugkörper besitzen hohe Anforderungen an die Zuverlässigkeit von Hardware und Software. Unter dem Einsatz von Ressourcen in redundanten Konstellationen, wird der geforderte Grad an Fehlertoleranz erreicht. Dabei existieren weitreichende Ansätze und Algorithmen zur Erkennung, Detektion, Isolation und Behebung von Fehlern, vergleiche [NAS12]. So werden einerseits die Eintrittswahrscheinlichkeiten und Auswirkungen von Fehlern

analysiert und andererseits Behandlungsstrategien zur Wiederherstellung des Betriebs implementiert.

Tritt im Betrieb ein Ressourcenfehler auf, kompensiert das System diesen durch einen Wechsel zu einer redundanten Ressource. Ist diese Redundanz nicht vorgesehen oder die entsprechende Alternative bereits defekt, erfolgt ein Übergang in einen Notmodus. Ab diesem Zeitpunkt wird das Eingreifen des Wartungspersonals durch einen Schnittstelle zur Fernwartung notwendig. Das Personal analysiert die Situation und versucht eine Rekonfiguration ohne Verwendung der defekten Bauteilgruppe festzulegen, vergleiche Beispiel aus Abschnitt 1.1.

Der vorgeschlagene Ansatz lässt sich in die Fehlerbehandlung integrieren. Dabei werden Beziehungen zwischen Bauteilgruppen unterschiedlichen Typs qualitativ analysiert und in der Alternativensuche berücksichtigt. Durch Restriktionen auf der strukturellen Ebene der Ressourcenplattform werden grundsätzlich nur valide Konfigurationen durch den Ansatz erhoben. Neben der Entscheidungsunterstützung in der Alternativeinauswahl, ist der Ansatz grundsätzlich auch für das autarke Treffen der Entscheidungen in das System integrierbar.

Zur detaillierten Validierung, vergleiche Abschnitt 6, wird in dieser Arbeit die Domäne der Weltraumsysteme näher betrachtet.

1.5. Durchgängiges Beispiel: Umfelderfassung eines Fahrerassistenzsystems

Mit der Einführung von Geschwindigkeitsregelanlagen (kurz: Tempomat[2]) in Kraftfahrzeugen, wurde erstmalig eine softwareintensive Regulierung der Fahrgeschwindigkeit im öffentlichen Straßenverkehr umgesetzt. Eine Geschwindigkeitsregelanlage realisiert die kontinuierliche Anpassung der Motordrehzahl, um eine durch den Fahrer vorgegebene Geschwindigkeit, unabhängig von Umwelteinflüssen, stabil zu halten. Die Regelung beachtet dabei unter anderem das Fahrverhalten anderer Verkehrsteilnehmer nicht.

[2] patentiert durch Daimler AG, `https://register.dpma.de/DPMAregister/marke/register/932012/DE`

So muss der Fahrer beispielsweise auf das Abbremsen eines vorausfahrenden Fahrzeugs selbst reagieren.

Eine Weiterentwicklung des Tempomaten bildet die adaptive Abstands- und Geschwindigkeitsregelung (Englisch: Adaptive Cruise Control, ACC). Ein ACC überwacht als Fahrerassistenzsystem das Umfeld vor dem Fahrzeug und kann autark auf Änderungen reagieren und die gewünschte Fahrgeschwindigkeit temporär reduzieren und wiederherstellen. Eine weitere Verwendung dieser Regelsysteme sind unter anderem Notbremsassistenzen und Unfallwarnsysteme mit Warnlichtautomatik. Die Literatur beschreibt zwei unterschiedliche ACC-Ausprägungen. In der Basisvariante regelt ein sogenanntes Standard-ACC [ISO10] die Geschwindigkeiten oberhalb von circa 30 km/h [Jon01]. Eine Regelung für langsame Fahrten bis zum Stillstand wird durch ein erweitertes ACC [ISO09] (Englisch: Full speed range adaptive cruise control, FSRA) realisiert. In den folgenden Abschnitten wird näher auf ein Standard-ACC eingegangen. Dabei sollen zunächst typische Betriebsanforderungen an solche Systeme skizziert werden.

1.5.1. Betriebsmodi eines Standard-ACC

Winner et al. fassen in ihrem Standardwerk [WHWSH15] die Anforderungen an ein Standard-ACC in den Betriebsmodi Freifahrt, Folgefahrt und Annäherung zusammen und definieren Funktionsgrenzen.

Eine *Freifahrt* beschreibt ein zum Tempomaten ähnliches Verhalten, wobei das System eine konstante Wunschgeschwindigkeit einhält und nur auf einen Bremseingriff oder andere Eingaben des Fahrers reagiert. Die Fahrt soll dabei mit hohem Komfort und ohne erkennbare Geschwindigkeitsabweichungen ablaufen. Die Regelung soll die übliche Fahrdynamik des Fahrers nachahmen. In der *Folgefahrt* wird die eigene Geschwindigkeit und ein eingestellter Abstand auf die eines vorausfahrenden Fahrzeugs kontinuierlich angepasst. Zur Steigerung des Fahrkomforts werden leichte Ungleichmäßigkeiten im Beschleunigungs- und Verzögerungsverhalten des Vorausfahrenden abgefedert. Eine automatische Objekterkennung im Abstandsbereich ermöglicht ein Reagieren auf das Ein- oder Ausscheren von Fahrzeugen. Beim Einscheren reagiert das System durch langsames Zurückfallen, um ein drastisches Abbremsen zu vermeiden. Eine *Annäherung* an ein Fahrzeug kann langsam oder schnell erfolgen. Im ersten Fall wird zügig der Sollabstand durch das

System hergestellt. Bei hoher Differenzgeschwindigkeit wird eine Verzögerung mit absehbarem Verlauf eingeleitet, die dem Fahrer den Bedarf eines manuellen Eingreifens erkennbar macht.

Das Standard-ACC unterliegt einigen Funktionsgrenzen, die die Übergabe zum Fahrer bedingen. So findet keine Regelung bei geringen Geschwindigkeiten statt. Eine Zeitlücke, das heißt der zeitliche Abstand zum vorausfahrenden Fahrzeug, darf eine Sekunde nicht unterschreiten. Der Fahrer behält die Kontrolle zur Übersteuerung der automatischen Regelung durch Gas- und Bremspedale. Das System muss bei einem Ausfall jederzeit noch eine geeignete Übergabe an den Fahrer ermöglichen. Zur Erfüllung dieser betrieblichen Anforderungen stützt sich ein ACC auf eine Vielzahl von Sensoren, Aktuatoren und berechnenden Ressourcen.

1.5.2. Aufbau eines Standard-ACCs

Der Aufbau eines Standard-ACCs lässt sich nach Winner et al. [WHWSH15] in 16 Funktionsmodule untergliedern, siehe Abbildung 1.3. Neben Modulen zur Bedienung, Anzeige und Diagnose, basiert der Hauptteil des Systems auf umfangreicher Sensorik zur Umfelderfassung und Aktuatorik zur Ansteuerung von Beschleunigung und Verzögerung des Fahrzeugs.

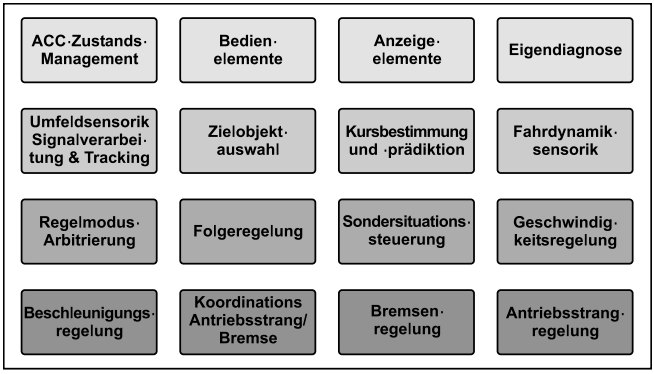


Abbildung 1.3.: Funktionsmodule eines ACCs (Quelle: [WHWSH15])

In dieser Arbeit dient ein Standard-ACC zur Veranschaulichung der Rekonfigurationsaspekte eines Systems mit inhärenter Ressourcenredundanz und qualitativen Degradierungen. Im Fokus ist die redundante Sensorik, die für eine Vielzahl von Systemausprägungen mit unterschiedlichen qualitativen Anforderungen zur Umfelderfassung genutzt werden kann.

Als eine technische Ausprägung eines Standard-ACC wird in Abbildung 1.4 der Aufbau des ersten marktreifen *Distronic*-Systems der Marke *Mercedes-Benz* aus dem Jahr 1999 gezeigt.

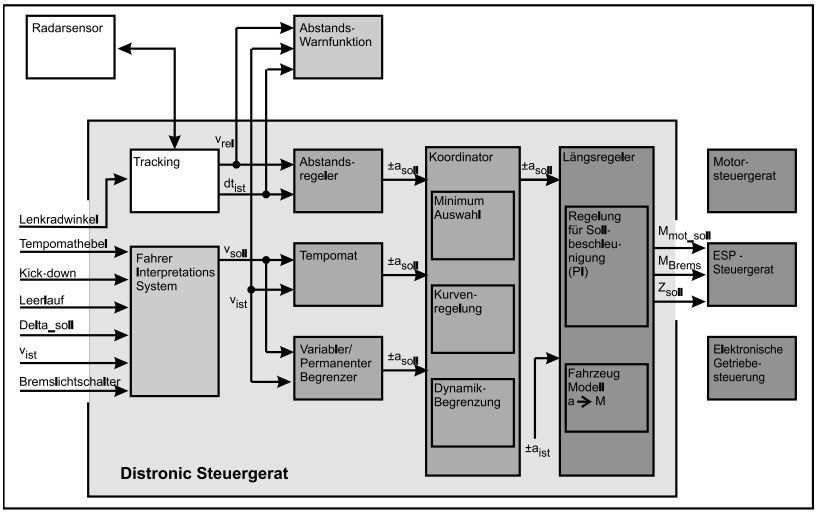


Abbildung 1.4.: Aufbau Distronic von Mercedes-Benz (Quelle: [WHWSH15])

Die Umfelderfassung (Englisch: *tracking*) erfolgt bezüglich der Stellung des Lenkrads sowie einer RADAR-Messung zur Ermittlung der relativen Geschwindigkeit (v_{rel}) und des zeitlichen Abstands zum vorausfahrenden Fahrzeugs (dt_{ist}). Ein Fahrerinterpretationssystem fasst sämtliche relevanten Fahreraktionen beziehungsweise Eingabesignale (zum Beispiel Istgeschwindigkeit V_{ist}, Stellung des Tempomaten, Sollabstand dt_{soll}) zusammen und ermittelt daraus eine Sollgeschwindigkeit V_{soll}. Der Tempomat ermittelt aus Ist- und Sollgeschwindigkeit einen Beschleunigungs-/Verzögerungswert $a\pm_{soll}$.

Ebenso ermitteln ein Abstandsregler und weitere Module den Wert $a\pm_{soll}$. Ein Koordinator führt unter Beachtung von Kurvenfahrten, Fahrdynamiken und Wertbeschränkungen die Aggregierung aller Werte von $a\pm_{soll}$ durch. Im Längsregler wird letztlich eine Senkung beziehungsweise Erhöhung der Motordrehzahl oder das Aktivieren der Bremsen in Abhängigkeit der Eingabewerte sowie eines Fahrzeugmodels gesteuert. Die Regelstrecke entspricht einer Reihe weiterer Steuergeräte, vorrangig dem ESP-Steuergerät.

Im Folgenden wird von der technischen Umsetzung eines ACCs abstrahiert. Die Funktionsmodule werden auf einen Teil der Sensorik zur Umfelderfassung und einen integrierten Block zur Abstandsreglung reduziert. Bei der Sensorik wird verstärkt auf deren Redundanz und Austauschbarkeit eingegangen.

1.5.3. Sensorik in ACC-Systemen

Zur Erfassung von Abständen kommen dabei in heutigen Automobilen vor allem zwei Sensorentypen zum Einsatz: RADAR und LiDAR. Die Messung bei RADAR-Systemen (Englisch: Radio Detection And Ranging) erfolgt über elektromagnetische Wellen die von einem zu beobachteten Objekt reflektiert werden. Im Automobilbereich werden Wellen in verschiedenen Frequenzmodulationen eingesetzt. Der Doppler-Effekt ermöglicht dabei die Ermittlung von Entfernung und Geschwindigkeit des Objekts. LiDAR-basierte Systeme (Englisch: Light Detection And Ranging) nutzen optische Messverfahren mit ultravioletten, infraroten oder sichtbaren Lichtstrahlen zur Entfernungsmessung.

Laut Winner et al. [WHWSH15] setzen europäische Automobilhersteller vorrangig die RADAR-Technik im Bereich der Standard-ACCs (ACC Basis) ein, vergleiche siehe Abbildung 1.5. Bei asiatischen Fabrikaten dominiert hingegen LiDAR als kostengünstige Alternative, zum Beispiel beim Automobilhersteller *Nissan*. Beide Systeme lassen sich nicht abschließend qualitativ abgrenzen, haben jedoch Vorteile je nach Betriebsmodus. Aus ökonomischen Gründen werden in heutige Fahrzeuge nicht beide Systemvarianten integriert. Im Zuge der stetigen Weiterentwicklung des autonomen Fahrens [MGLW15] nimmt auch die Zahl und Ausprägung an Sensoren zur präzisen und zuverlässigen Umfelderfassung zu. In dieser Arbeit soll ein

experimentelles Fahrzeug aus der Forschung mit einer umfassenden publizierten technischen Dokumentation als Vorlage für ein *durchgängiges Beispiel* dienen.

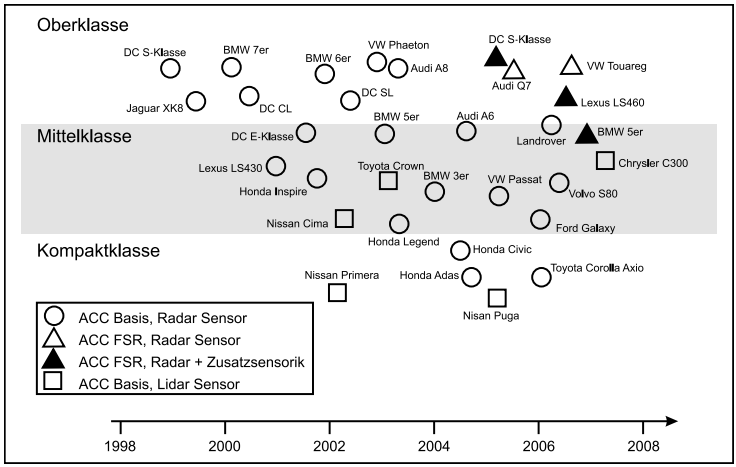


Abbildung 1.5.: Marktübersicht zur ACC-Sensorik in Fahrzeugen unterschiedlicher Klassen (Quelle: [WHWSH15], VDA Arbeitskreis ACC AK3.11)

1.5.4. Forschungsträger Leonie (Stadtpilot)

Das Forschungsprojekt *Stadtpilot* der Technischen Universität Braunschweig untersucht das autonome Fahren in stark befahrendem Stadtgebiet[3]. Das vordergründige Forschungsziel liegt in einer vollständigen Umrundung der Ringstraße um die Braunschweiger Innenstadt. Dies beinhaltet im Einzelnen das Einfädeln in fließenden Verkehr, die Fahrstreifenerkennung, der Fahrstreifenwechsel, das Passieren von Kreuzungen mit und ohne Lichtsignalanlagen sowie das Einparken. Sämtliche Aktivitäten enthalten die ständige Berücksichtigung anderer Verkehrsteilnehmer. Nothdurft et al. [NHO+11] geben einen ausführlichen Überblick über das Projekt Stadtpilot und dessen gesetzlichen Kontext.

[3] `https://www.tu-braunschweig.de/stadtpilot`

In dem Projekt Stadtpilot wurde als Versuchsträger der autonom fahrende PKW *Leonie* auf Basis eines Volkswagen Passat Variant aufgebaut. Durch den Einsatz einer Vielzahl von Sensorik zur Umfelderfassung, Aktuatorik zur automatischen Fahrtensteuerung und leistungsstarke Rechenressourcen wird die autonome Fahrt realisiert. Im weiteren Verlauf soll näher auf die Sensorik in der Fahrzeugfront eingegangen werden, mit einem besonderen Augenmerk auf das ACC-System. Ohl fasst in seiner Arbeit [Ohl14] die Sensorik wie folgt zusammen: Die Abbildung 1.6 zeigt die Front des Versuchsträgers, wobei alle Umfeldsensoren hervorgehoben und nummeriert wurden.

Abbildung 1.6.: Sensoren am Versuchsträger Leonie (Quelle: [Ohl14])

An den Außenecken des Stoßfängers befindet sich je ein LiDAR-Sensor des Typs *IBEO Alaska XT* (1). Unterhalb des Kennzeichens ist ein *SMS UMMR 2010* RADAR-Sensor (2) sowie zwei Multi-Beam-LiDAR-Sensoren von den Typen *Hella IDIS* und *Hella IDIS 2* (3, nebeneinander). Pro Fahrzeugseite ist ein LiDAR-Sensor von der Bauart *Sick LMS100* zur Fahrstreifenerkennung (4) montiert. Auf dem Fahrzeugdach wurde mit dem *Velodyne HDL64ES2* ein rotierender LiDAR-Sensor (5) hinzugefügt.

Im Heck des Fahrzeugs sind weitere Sensoren verbaut. Für die Umfelderfassung des ACCs nutzt das System ausschließlich die beiden IDIS-Sensoren

und den UMMR-Sensor. Aus diesem Grund wird in dieser Arbeit nicht auf weitere Sensoren näher eingegangen.

Die drei Sensorentypen besitzen die folgenden technischen Spezifikationen, vergleiche [Ohl14]. Der LiDAR-Sensor IDIS1 erfasst Gegenstände in einem Bereich von *12 Grad* bei einer Auflösung von 1,0 Grad. Die maximale Reichweite der Erfassung beträgt *200 m* bei einem Messfehler von ±(1% + 1 m). Als Nachfolger des Sensors erweitert der IDIS2 den Öffnungswinkel auf *161 Grad*, senkt jedoch die Reichweite auf *150 m*. Die Auflösung sinkt auf 1,8 Grad, der Messfehler bleibt jedoch konstant zum Vorgänger. Zusätzlich ermöglicht der Sensor die Erfassung der Geschwindigkeit auf bis zu 1 km/h genau, vergleiche [WHWSH15].Der RADAR-basierte Sensor UMMR 2010 misst in einem Öffnungswinkel von *18 Grad* bei *160 m* Reichweite. Der Messfehler unterhalb von 10 m Abstand liegt bei ±0,25 m, darüber bei ±2,5%. Die Geschwindigkeit kann in Schritten von ±0,25 km/h gemessen werden.

Zur Abstraktion des existierenden umfangreichen ACC-Systems mit Anbindung an die autonomen Funktionalitäten von Leonie, wird für diese Arbeit ein vereinfachtes System in Anlehnung an das Distronic-ACC, siehe Abbildung 1.4, entworfen. Die Systemübersicht in Abbildung 1.7 zeigt die Abhängigkeit des Trackings von den drei Umfeldsensoren, sowie die Interpretation der Fahrereingaben (Gas, Bremse, Lenkrad, Sollabstand) und Fahrzeugwerte (Istgeschwindigkeit V_{ist}, Istbeschleunigung $\pm a_{ist}$).

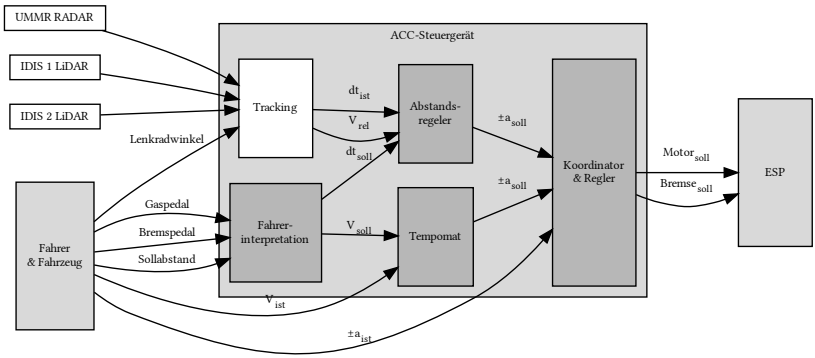


Abbildung 1.7.: Systemübersicht eines ACCs mit RADAR und LiDAR

Das Tracking ermittelt die relative Geschwindigkeit V_{rel} und den zeitlichen Abstand dt_{ist}. Aus diesen Daten und dem Sollabstand dt_{soll} ermittelt der Abstandsregler eine notwendige Beschleunigung oder Verzögerung $\pm a_{soll}$. Der Tempomat ermittelt ebenfalls einen Wert $\pm a_{soll}$ aus gewünschter und tatsächlicher Fahrtgeschwindigkeit V_{soll} beziehungsweise V_{ist}. Der kombinierte Koordinator und Längsregler ermittelt aus vorliegendem Beschleunigungs- oder Verzögerungswert $\pm a_{ist}$ die notwendige Verzögerung oder Beschleunigung ($Bremse_{soll}$ und $Motor_{soll}$) des Fahrzeugs.

1.5.5. Rekonfigurationsraum

Der Raum möglicher Systemkonfigurationen des durchgängigen Beispiels ist als Feature-Modell [KCH+90] in Abbildung 1.8 dargestellt.

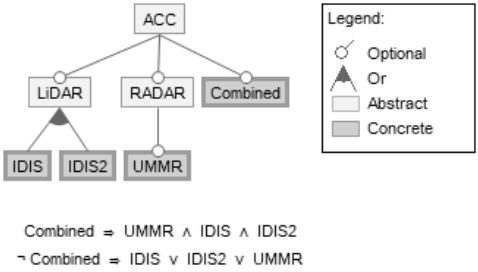


Abbildung 1.8.: Feature-Modell des durchgängigen Beispiels

Die Features repräsentieren im vorliegenden Modell die Verwendung konkreter Ressourcen. So können Konfigurationen von einer beliebigen bis zu allen möglichen Ressourcen frei gebildet werden. Weiterhin bedingt die Selektion des Features „combined“die gleichzeitige Auswahl aller Ressourcen. In dieser Arbeit werden vier Systemkonfigurationen näher betrachtet.

In dem durchgängigen Beispiel werden somit unterschiedliche Sensortypen eingesetzt. Zur Anwendung des Ansatzes ist es zu prüfen, ob diese dennoch Überschneidungen in der Funktionalität zulassen. Cacilo et al. [CSW+15] beschreiben den technologischen Unterschied von RADAR und LiDAR anhand

einer Liste von Vor- und Nachteilen in Abhängigkeit der Umweltbedingungen.

Ein RADAR-System weist sich durch eine hohe Robustheit gegenüber Wettereinwirkung, Verschmutzung und Lichteinstrahlung sowie der genauen Messung von Distanz und relativer Geschwindigkeit aus. Weiterhin können solche Systeme durch Kunststoff verdeckt einbaut werden und benötigen einen geringen Bauraum. Die Nachteile von RADAR bestehen in der groben Auflösung, einem relativ kleinem Erfassungsbereich und Ungenauigkeiten in der Objektklassifizierung. Weiterhin können elektromagnetische Störungen auftreten und Phantomobjekte erkannt werden.

Die LiDAR-Technik bietet eine feine Auflösung und hohe Genauigkeiten bei großer Reichweite. Neben der Objekterkennung ist eine präzise Bestimmung der Relativgeschwindigkeit möglich. Die Erfassung ist sehr schnell und unabhängig vom Tageslicht möglich. Beim Einsatz von rotierenden Sensoren ist zusätzlich ein großer Öffnungswinkel gegeben. Die Sensorik ist jedoch anfällig gegenüber Regen und Gegenlicht. Nebel oder Staub reduzieren die Reichweite zudem massiv. Die Erfassung von Objekten, die Licht stark spiegeln, durchlassen oder nicht reflektieren, ist nur eingeschränkt möglich. Insbesondere die Beschaffenheit der Fahrbahn beeinflusst die Rückstreuung des Lichts und damit die Messqualität. Weiterhin ist eine Klassifizierung von Objekten nur ungenau möglich. Rotierende Sensoren benötigen viel Bauraum und sind wegen beweglicher Teile störungsanfällig. Je nach Umweltbedingung und Benutzungsmodus liegen Überschneidungen vor.

1.5.6. Qualitätsdimensionen

Die Systemkonfigurationen werden unter anderem durch die vier ressourcenabhängigen Qualitätsdimensionen *Öffnungswinkel*, *Reichweite*, *Robustheit* im Sinne der Störungsunempfindlichkeit und *Verfügbarkeit* durch Ressourcenredundanz charakterisiert. In Tabelle 1.1 sind die Dimensionen mit Werten belegt, die sich aus den technischen Unterlagen[4] der Sensoren ableiten. Eine Besonderheit bildet die Systemkonfiguration C_3. In dieser Konfiguration leiten sich die Belegungen für die ersten drei Dimensionen aus dem Mittelwert

[4] Datenaufbereitung von Ohl [Ohl14] mit ergänzenden Information von Cacilo et al. [CSW+15].

aller verwendeten Ressourcen ab. Die vierte Dimension (Verfügbarkeit) basiert hingegen auf der Summe der individuellen Ressourcenverfügbarkeiten, da alle Ressourcen zugleich eingesetzt werden.

Tabelle 1.1.: Qualitätsdimensionen und Wertebelegungen der Konfigurationen des durchgängigen Beispiels

Konfiguration	Ressoucen	Winkel	Reichweite	Robustheit	Verfügbarkeit
C_0	IDIS2	**161**	150	0,5	0,5
C_1	IDIS1	12	**200**	0,4	0,4
C_2	UMMR	18	160	**0,9**	0,9
C_3	IDIS1, IDIS2, UMMR	63,7	170	0,6	**1,8**

1.5.7. Degradierung

Zur Kompensation partieller Sensorikausfälle, werden Degradierungsstufen in den Systementwurf integriert. Nach Winner et al. [WHWSH15] kann so das Ausfallen eines Nahbereich-RADAR durch ein Fernbereich-RADAR kompensiert werden, wenn zugleich der Betriebsmodus in ein Standard-ACC wechselt. Somit sind zumindest Messungen oberhalb einer Mindestgeschwindigkeit noch möglich. Eine wesentliche Anforderung an die Degradierung von beobachtbaren Funktionalitäten ist, dass die Änderungen dem Fahrer im Fahrzeug transparent dargelegt werden müssen. Ist diese nicht möglich, ist anstelle einer unplausiblen Erklärung, die Funktionalität vorzugsweise abzuschalten.

Eine detaillierte Analyse der Degradierungen in dem durchgängigen Beispiel erfolgt in Abschnitt 3.2.

1.6. Gliederungsüberblick

Die Arbeit ist in sieben Kapitel unterteilt und wie folgt aufgebaut.

Kapitel 1 führt die Zielsetzungen für den anvisierten Ansatz entlang von vier Forschungsfragen ein. Weiterhin wird eine Einordnung vorgenommen sowie Kontextannahmen und drei mögliche Domänen zur Anwendung des Ansatzes vorgestellt. Abschließend wird ein durchgängiges Beispiel aus der Automotive-Domäne zur Illustration des Ansatzes eingeführt.

In **Kapitel 2** werden die zum Verständnis dieser Arbeit notwendigen Grundbegriffe ausgelegt und eine Einführung in aufgegriffene Ansätze und Techniken gegeben. Begrifflichkeiten der modellbasierten Entwicklung, Wartung und Fehleranalyse stehen dabei im Vordergrund. Die aufgegriffenen Ansätze behandeln die Gebiete komponentenbasiertes Software Engineering, Variabilitätsmodellierung, multikriterielle Optimierung, Architekturevaluation, Selbstmanagement und empirische Studien.

Kapitel 3 führt den dreiphasigen Prozess zur Entscheidungsunterstützung für Wartungsaufgaben ein. Dabei werden die einzelnen Phasen von der Vorausberechnung des Entwurfsraums sowie die Relationsbildung von Architekturen im Entwurf bis hin zur Fehlerbehandlung im Betrieb detailliert vorgestellt. Das Kapitel liefert einen Überblick und verweist auf die folgenden drei Hauptkapitel der Arbeit.

Das **Kapitel 4** bildet das erste Hauptkapitel und führt die zugrundeliegende Graphstruktur Architekturrelationsgraph (*ARG*) für das Expertensystem zur Entscheidungsunterstützung ein. Dabei werden zunächst die strukturellen Modelle formal definiert und in Metamodell überführt. Im Anschluss erfolgt eine algorithmische Beschreibung der Instanziierung und Parametrisierung der Modelle.

Kapitel 5 ist das zweite Hauptkapitel und beschreibt die Erzeugung des deterministischen Architekturrelationsgraphens (*DARG*) auf Grundlage der zuvor eingeführten Strukturen und erzeugenden Algorithmen für den kostenoptimalen *ARG* und den Qualitätsattributsgraphen (*QADAG*). Dabei wird detailliert beschrieben wie durch Vorausberechnungen zur Entwurfszeit die Identifikation konkreter Betriebsmodi adäquate betriebsoptimale Rekonfigurationsalternativen zur Laufzeit ermöglicht wird. Die Anwendbarkeit der

Analyse zu unterschiedlichen Zeitpunkten im Entwicklungs- und Wartungsprozess wird weiterhin thematisiert. Abschließend erfolgt eine Beschreibung einer Erweiterung des Ansatzes zur Berücksichtigung von variierenden Betriebsmodi.

Das **Kapitel 6** dokumentiert als letztes Hauptkapitel die Validierung zur Beurteilung der Effektivität des Ansatzes zur effizienten Entscheidungsunterstützung in der Wartung. Dabei werden vorrangig die Verringerung der Aufwände in der Kommunikation und den Tätigkeiten des Wartungspersonals betrachtet. Einleitend wird das strukturierte Vorgehen und die generellen Zielsetzungen der Validierung beschrieben. Im Kern des Kapitels stehen 17 Validierungsfragen, die einerseits in einer werkzeuggestützten Messung und anderseits in einer empirischen Studie untersucht werden. Neben den Messungen auf Basis der eingeführten Modelle wurde ein großes Gewicht auf die Beurteilung und Akzeptanz innerhalb der gewählten Domäne gelegt. Das Kapitel beschreibt dies entsprechend umfangreich. Zur realitätsnahen Untersuchung beider Teilaspekte wurde als Anwendungsszenario das Lageregelungssystem eines existierenden Mikrosatellitens herangezogen. Die Validierung schließt mit einer Beurteilung der Zielerreichung der Validierungsfragen ab. Das Kapitel zeigt weiterhin die Stabilität des Ansatzes im Rahmen der Erweiterung zur Berücksichtigung variierender Betriebsmodi. Als Abschluss führt das Kapitel die gesammelten Erkenntnisse aus allen Untersuchungen zur Beantwortung der Forschungsfragen zusammen. Dabei werden abschließend Limitierungen und mögliche Erweiterungen diskutiert.

Das **Kapitel 7** gibt zunächst einen Einblick in die Rolle des modernen Software Engineerings innerhalb der Domäne der Weltraumsysteme. Weiterhin werden Arbeiten aus den Bereichen variantenreiche Systemmodellierung, metaheuristische Exploration und Distanzmaßbildung in Entwurfsräumen, Entscheidungsunterstützung für Rekonfigurationen und Dokumentationsansätze für Wartungsprozesse in Bezug zu dieser Arbeit gesetzt.

Im letzten **Kapitel 8** erfolgt eine zusammenfassende und kritische Betrachtung der erzielten Ergebnisse. Weiterhin wird ein ein Ausblick auf mögliche Anschlussarbeiten gegeben.

2. Grundlagen

Zum Verständnis dieser Arbeit werden die notwendigen Grundbegriffe ausgelegt und eine Einführung in aufgegriffene Ansätze und Techniken gegeben.

2.1. Begriffliche Festlegungen

Im Folgenden werden Begrifflichkeiten, die für diese Arbeit relevant sind, auf das behandelte Thema eingegrenzt und charakterisiert.

2.1.1. Systembegriff

Ein *software-intensives technisches System* wird durch zwei Bestandteile charakterisiert.

Software-intensives System: Gemäß ISO/IEC/IEEE 42010:2011 ist ein *software-intensives System* definiert als ein System, in dem Software einen wesentlichen Einfluss auf Entwurf, Konstruktion, Deployment und Evolution des Gesamtsystems hat. Dabei können sowohl Einzelanwendungen, traditionelle (Sub-)Systeme, Produktlinien, Produktfamilien als auch „Systems of Systems“ software-intensiv sein.

Technisches System: Ein *Technisches System* fasst in Bezug auf die Ingenieurwissenschaften laut Brockhaus[1] „beliebige technische Hervorbringungen“ unter einem ganzheitlichen Begriff zusammen. Somit werden mehrfach definierte Begriffsausprägungen wie beispielsweise Maschine, Gerät oder

[1] `https://brockhaus.de/ecs/enzy/article/technik-20/technische-systeme`

Apparat unter einem Sammelbegriff eingeordnet. Nach Czichos [Czi15] können technische Systeme dabei in drei Kategorien unterteilt werden. Die materialbasierten technischen Systeme adressierten Produktionsanlagen oder Transportsystemen, die Stoffe unter anderem gewinnen, bearbeiten oder transportieren. Energiebasierte technische Systeme haben die Aufgabe Energie zu erzeugen, umzuwandeln oder zu nutzen. Beispiele dafür sind Generatoren oder Antriebssysteme. Ein informationsbasiertes technisches System, zum Beispiel ein Smartphone oder Multimediagerät, erzeugt, überträgt oder zeigt Informationen.

2.1.2. Architekturbegriff

Eine *Architektur* beschreibt nach ISO/IEC/IEEE 42010:2011 die inneren Strukturen eines Software-/Hardwaresystems. Dabei werden neben der Definition von Komponenten, deren Beziehungen zueinander und gegenüber der Umgebung definiert.

Software- und Hardwarearchitektur: Für detaillierte Betrachtungen lässt sich eine Systemarchitektur in die Bestandteile *komponentenbasierte Softwarearchitektur* und *Hardwaretopologie* aufteilen. Erstere beschreibt die funktionale Struktur und die Beziehungen von Softwarekomponenten, letztere die technischen Aspekte der Hardwareressourcen. Im weiteren Verlauf dieser Arbeit werden primär komponentenbasierte Softwarearchitekturen betrachtet und dabei verkürzt als Architektur bezeichnet.

Softwarekomponente: Eine *Softwarekomponente* (hier: Basiskomponente) beschreibt eine atomare Entität einer Softwarearchitektur, die eine Teilfunktionalität des Systems kapselst. Schnittstellen definieren den Informationsaustausch mit der Umgebung. Softwarekomponenten können hierarchisiert in Softwarekompositionen auftreten.

Hardwareressource: Als *Hardwareressource* wird ein elektronisches Bauteil angesehen, welches die Ausführung der Software ermöglicht. In eingebetteten Systemen treten Ressourcen zur Datenverarbeitung (Prozessoren,

Speicher) auf und ebenfalls als Sensorik (Datenquellen) und Aktuatorik (Datensenken).

Im untersuchten Ansatz liegt der Fokus auf der Rekonfiguration komponentenbasierter Softwarearchitekturen. Von der Hardwaretopologie wird dabei weitestgehend abstrahiert. Die Topologie wird konkret als *Ressourcenplattform* angesehen, die in Untermengenbeziehungen von unterschiedlichen Ausprägungen einer Softwarearchitektur angefordert wird.

2.1.3. Konfigurationsbegriff

Eine *Konfiguration* eines Systems beschreibt die Selektion von Soft- und Hardwareelementen sowie deren Beziehungen zueinander. Die inneren Strukturen der Soft- und Hardware werden in Konfigurationen nicht beschrieben. Laufzeitdaten (Variablenbelegungen) sind hier nicht Teil einer Konfiguration. Somit bleibt eine Konfiguration im Betrieb strukturell unverändert. Eine Anpassung von Software oder Hardware bedarf dem Wechsel in eine neue Konfiguration.

Rekonfiguration: Der Wechsel von einer Konfiguration zur nächsten wird als *Rekonfiguration* bezeichnet. Es handelt sich um den Prozess der Umgestaltung der Soft- und Hardwarestruktur eines Systems. Rekonfigurationen finden bereits zur Entwurfszeit in der Erprobung von Implementierungsvarianten statt, sind jedoch auch zur Laufzeit möglich. Letzteres erfordert ein aufwendiges Management von Laufzeitdaten. Für den hier untersuchten Ansatz stehen Laufzeitaspekte im Vordergrund, sowie durch Anwendung von Rekonfigurationen zur Reaktion auf Fehler in der Hardware. Der Erhalt einer möglichst gleichbleibenden Funktionalität ist in diesem Ansatz das maßgebliche Ziel. Dabei ist eine schwankende Qualität (*Degradationen*) der Funktionalität des Systems dennoch akzeptabel.

Adaptation: Die Laufzeitadaptation beschreibt eine Erkennung des Laufzeitkontextes eines Systems und die Anpassung durch autonome Rekonfigurationen. Dabei sind nachträgliche Erweiterungen des Systemverhaltens möglich, zum Beispiel auf Basis veränderter Datenflüsse im System.

In dieser Arbeit wird auf vorberechnete Konfigurationsmengen und deren optimierte Übergangsbeziehungen zurückgegriffen. Ein Adaptation findet nicht statt. Die *Betriebsszenarien* des Systems sind bekannt und eine Rekonfiguration bleibt für den Anwender durch festgelegte Übergangsregeln kontrolliert.

2.1.4. Entwurfsraumausprägungen

Der Entwurfsraum beschreibt das Universum möglicher Konfigurationen, die strukturell und funktional valide zu auszubilden sind.

Vorselektierter Entwurfsraum: In dieser Arbeit wird in der Analyse zur Entwurfszeit auf eine qualitative Vorselektion gültiger Konfigurationen zurückgegriffen. Der vorselektierte *Entwurfsraum* bezieht sich hier somit bereits auf eine echte Teilmenge der möglichen Konfigurationen.

Rekonfigurationsraum: Als *Rekonfigurationsraum* wird hier eine Untermenge des vorselektierten Entwurfsraums, in der Beziehungen zwischen Konfigurationen explizit gemacht sind. Die Untermenge wird entlang einer Begrenzung von Rekonfigurationsaufwänden gebildet und anschließend qualitätsorientiert sortiert.

2.1.5. Softwarewartung

Technische Maschinen unterliegen Verschleiß, stark belastete Bauteile nutzen im Betrieb ab. Maschinen sind daher regelmäßig zu warten, um die Alterung der gesamten Anlage durch Austausch von Bauteilen zu verlangsamen. Vorrangig werden dabei Bauteile mit identischer Funktion ein- und ausgebaut. Software altert zwar auch relativ zu seiner Ausführungsplattform, eine Wartung ist jedoch als Rekonfiguration vorhandener Bestandteile zu verstehen. Dies kann sich beispielsweise durch eine *qualitative Degradation* (zum Beispiel Kompatibilitätsmodus oder abgesicherter Betriebsmodus) darstellen.

Der Begriff *Softwarewartung* wird durch ISO/IEC 14764:2006 unterteilt in die *korrektive*, *optimierende*, *adaptive* und *erweiternde Wartung*.

Lientz et al. [LST78] charakterisieren diese Begriffe im Rahmen von *konservierende und progressiven Wartungsaktivitäten*.

Konservierende Wartungsaktivitäten: Sowohl die korrektive als auch die optimierende Wartung zählen zur ersten Kategorie. Einerseits werden Fehlerbehandlungen und andererseits Maßnahmen zur Steigerung der Softwarequalität (zum Beispiel Refactoring) durchgeführt.

Progressive Wartungsaktivitäten: Progressive Aktivitäten treten hingegen in der adaptiven und erweiternden Wartung auf. Neben Migrationen werden dabei Maßnahmen zur Erweiterung der Funktionalität umgesetzt.

Unmittelbare und verzögerte korrektive Wartung: Die korrektive Wartung von Software lässt sich nach Bommer et al. [BSB08] weiter untergliedern. Die *unmittelbare korrektive Wartung* (Englisch: immediate corrective maintenance) versucht kurzfristig nach Auftreten eines Fehlers eine Reparatur durchzuführen. Hingegen überprüft die *verzögerte korrektive Wartung* (Englisch: deferred corrective maintenance) im Fehlerfall zunächst die Erfüllung festgelegter Wartungsvorschriften und verhindert damit gegebenenfalls inadäquate Ad-hoc-Reparaturen.

Der hier vorgestellte Ansatz soll eine unmittelbare korrektive Wartung mit einem *selbst-adaptiven Wartungsmechanismus* unterstützen und somit sonst übliche Verzögerungen durch manuelle Analysen des Lösungsraumes verhindern. Gleichzeitig sollen Reparaturentscheidungen systematisch fundiert und nachvollziehbar sein.

2.1.6. Wartbarkeit

Wartbarkeit wird durch entsprechende Optimierungen in der Grundstruktur einer Maschine oder eines Softwareprodukts gesteigert. Im Maschinenbau steht hier vor allem die Zugänglichkeit und Austauschbarkeit von stark belasteten Bauteilen im Vordergrund.

Softwarewartbarkeit: Im Software Engineering ist Wartbarkeit gemäß ISO/IEC/IEEE 24765:2017 die Leichtigkeit der Modifikation eines Softwaresystems oder einer Komponente. Eine Modifikation definiert sich als Fehlerbehebung, Steigerung der Leistung oder andere Qualitätsattribute oder Anpassung an ein verändertes Betriebsumfeld. eines Softwaresystems oder einer Komponente. Im Softwareentwurf wird Wartbarkeit im ganzheitlichen Sinne durch flexible Strukturen gefördert, vergleiche [RSHR15, RHBR17]. Softwarewartbarkeit fokussiert sich nicht auf einzelne Bestandteile, sondern auf das gesamte Produkt. Dies lässt sich vor allem dadurch begründen, dass stark belastete Elemente nicht identifizierbar sind und der Alterungseinfluss oft unbekannt ist.

Inhärente Wartbarkeit langlebiger Softwaresystemen: Die Langlebigkeit eines Softwaresystems hängt von der Wartbarkeit ab. Bereits in frühen Arbeiten wurden Richtlinien [MKMG97] und Muster [BHS07] für wartbare Architekturen vorgeschlagen [MKMG97]. Moderne Ansätze, vergleiche [KDGR13], adressieren eine kontinuierlicher Softwarewartung als Teil der geführten Softwareevolution [GRG+14].

2.1.7. Fehleranalyse

Ein Fehler in einem System wird durch drei Begriffe charakterisiert, die sich in aufbauender Reihenfolge bewirken.

Fehlerursache: Eine Fehlerursache (Englisch: Fault) liefert die Begründungen für einen inkorrekten Systemzustand. Beispiele für Fehlerursachen sind konzeptionelle Entwurfsfehler oder Bedienungsfehler.

Fehlerzustand: Ein Fehlerzustand beziehungsweise Fehlzustand (Englisch: Error) beschreiben die exakte Position einer Abweichung von dem Sollverhalten als Artefakt, zum Beispiel falsche Attributsbelegung.

Versagen: Das Versagen (Englisch: Failure) ist die äußerlich erkennbare Auswirkung beziehungsweise Folge eines Fehlzustands, beispielsweise eine Falschausgabe.

Zur frühzeitigen Identifikation möglicher Fehlerursachen haben sich Fehleranalysemethoden etabliert, um das Risikomanagement im Entwicklungsprozess für fehlertolerante Systeme zu unterstützen.

Fehleranalysemethoden: In den Entscheidungsprozessen in der Wartung trägt das Risikomanagement eine entscheidende Rolle. Zur Fehler- und Auswirkungsanalyse werden etablierte Methoden eingesetzt. Hauptsächlich treten dabei Fehlerbäume (*FTA*, Englisch: Fault Tree Analysis), die Fehlermöglichkeits- und -einflussanalyse (*FMECA*, Englisch: Failure Mode and Effects Analysis) und Bayes'sche Netze in Erscheinung [CPNMH18]. Erkenntnisse aus den Analysen der Entwurfszeit der sicherheitskritischen Systemen werden zur Risikobeurteilung zur Laufzeit herangezogen.

2.1.8. Fehlerbehandlung

Die Behandlung von Fehlern erhöht die Fehlertoleranz eines System. *Fehlertoleranz* ermöglicht die Fortsetzung des produktiven Betriebs auch dann, wenn logische Fehler auftreten [Avi67].

Erkennung, Isolation und Behebung von Fehlern: Fehlertolerante Systeme integrieren Mechanismen, die auftretende Fehler im System lokalisieren, isolieren und den Betrieb wiederherstellen (*FDIR*, Englisch: Fault Detection Isolation Recovery). Im Bereich der Luft- und Raumfahrt liegen laut Zolghadri et al. [ZHC+14] aus Prozesssicht zwei Behandlungsstrategien vor, um kritische Ausfälle zu vermeiden.

Fehlertolerante Steuerung: Die *fehlertolerante Steuerung* (FTC, Englisch: Fault-tolerant Control) versucht eine Kompensation eines Fehlers zu ermitteln, wobei qualitative Degradationen vermieden werden. FTC bietet keine optimale Systemleistung, kann jedoch Fehlereffekte vermeiden. *Passive FTC-Ansätze* bieten eine gleichbleibende Stabilität und Leistung des Systems im

fehlerfreien und fehlerbehafteten Fall. Dabei wird die Robustheit des Systems ab dem Entwurf überladen, um mögliche Fehler ohne Auswirkungen zu kompensieren. Bei *aktiven FTC-Ansätze* beschreiben Reconfigurationsregeln, welche Maßnahmen im Fehlerfall durchgeführt werden müssen, um die Stabilität weiterhin zu gewährleisten.

Fehlertolerante Führung: Ansätze im Bereich der *fehlertoleranten Führung* (FTG, Fault-tolerant Guidance) bieten einen hohen Grad an Flexibilität zur sicheren Wiederherstellung eines stabilen Systemzustands. Dabei werden Kompensationsmechnismen an Bord des Flugkörpers genutzt. Nachdem ein Fehler erkannt und bestätig wurde, kann das System ohne eine weiteres Eingreifen des Wartungspersonals diesen kompensieren. In diesem Ablauf sind qualitativen Degradationen erlaubt. Eine Fortsetzung des produktiven Betriebs ist dann nur bei Änderungen der Missionsziele möglich. In dieser Arbeit wird ein Ansatz im Bereich der fehlertoleranten Führung verfolgt, der mögliche Lösungsstrategien vorschlägt, dabei jedoch mögliche Degradationen erforderlich macht.

Neben der Identifikation von Fehlern, sind auch redundanzbasierte Mechanismen zur Fehlerkompensation im fehlertoleranten Systementwurf integriert. Dies wird einerseits durch Vervielfachung von Systembestandteilen erreicht, andererseits durch Substitution der Funktionalität. Markley et al. [MC14] beschreiben diese zwei Klassen wie folgt.

Physikalische Redundanz: *Physikalische Redundanzen* oder auch Hardwareredundanzen beschreiben den physikalischen Austausch *homogener Ressourcen*. Dies erfordert im System die notwendige Kapazität zur Vervierfachung einzelner Bauteile. Werden zugleich mehrere homogene Ressourcen betrieben, sind diese *heiß-redundant* zueinander. Findet der Betrieb nur alternativ zueinander statt, liegen *kalt-redundante* Gruppierungen vor. Redundanzkonstellationen treten sequenziell in Blöcken oder überkreuzt zueinander auf. In Blöcken entstehen eindeutige Wirkungsketten von Sensorik über Datenverarbeitung hin zu Aktuatorik. Die Pfade in diesen Blockdiagrammen sind mit linearen Aufwand in Fehleranalysen zu überprüfen. Sind Ressourcen allerdings zueinander überkreuzt, zum Beispiel ein Sensor optional auf Datenverarbeitung 1 oder 2 geschaltet, liegt ein hoher Freiheitsgrad

der konkreten Redundanz vor. Der Aufwand zur Überprüfung aller Pfade steigt exponentiell.

Analytische Redundanz: *Analytische Redundanz* beschreibt hingegen Austauschbeziehungen zwischen einzelnen *heterogenen Ressourcentypen* oder Mengen solcher. So werden defekte Ressourcen durch ähnliche Bestandteile substituiert, wobei die geforderte Funktionalität zumindest beschränkt nachgebildet wird. Eine Anwendung dieser Methode findet sich beispielsweise im Bereich der Sensorfusion. In der Datenhaltung eines Systems sind analytische Redundanzen durch Fehlerkorrektur-Codes (Englisch: ECC, Error-correcting code), welche die Rekonstruktion von Teildaten ermöglichen, abgesichert (Informationsredundanz). Ein Beispiel für dieses Verfahren findet sich in ECC-abgesichertem Arbeitsspeicher. Zuletzt können Aktionen wiederholt angewandt werden, bis es zu einem Erfolg kommt (zeitliche Redundanz), z.B. bei der Erhebung eines Messwertes.

2.2. Grundlegende Konzepte und eingesetzte existierende Techniken

Unter Verwendung von Konzepten und Techniken aus den Bereichen komponentenorientierte Softwareentwicklung, Variantenmodellierung und multikriterieller Optimierung wird der Ansatz in dieser Arbeit entwickelt. Zum weiteren Verständnis werden notwendige Konzepte des Selbstmanagements von Softwaresystemen eingeführt. Im Rahmen der Validierung werden Kenntnisse aus dem Bereich der empirischen Studien benötigt, welche abschließend in diesem Kapitel erläutert werden.

2.2.1. Erstellung und frühzeitige Analyse komponentenorientierter Systemmodelle

Ein Großteil der bisherigen Forschung beschäftigt sich mit der initialen Systementwicklung, während der Betrieb und die Wartung zumeist davon

entkoppelt betrachtet werden. Der strategische Einsatz modellbasierter Bewertungen als Instrument zur Entscheidungsunterstützung für die Rekonfigurationen eines Systems zur Laufzeit, könnte beide Welten miteinander verknüpfen. Somit wäre auch eine autonome Fehlerbehebung von eingebetteten Systemen im Betrieb bedingt adressierbar.

Architektur Beschreibungssprachen

Modellbasierte Ansätze finden in den letzten Jahren verstärkten Einzug in die Entwicklungsprozesse in der Raumfahrtdomäne,vergleiche Abschnitt 7.1.1. Der Hauptteil der Betriebssoftware von Raumflugkörpern ist derzeit noch auf hochoptimierte Quelltexte zentriert. Der Ansatz in dieser Arbeit abstrahiert von technischen Details und ist dem Bereich der Optimierung komponentenorientierter Modelle für eingebettete Systeme zuzuordnen [GLB+07]. Im komponentenorientierter Software Engineering existieren eine Vielzahl von Arbeiten zur Architekturoptimierung [ABG+13a]. Architektur Beschreibungssprachen (ADL, Englisch: Architecture Description Language) ermöglichen die integrierte Modellierung von Systemstruktur und Analyseaspekten, vergleiche [RH08]. Analysetische und simulative Verfahren nutzen dieses Wissen zur frühzeitigen Beurteilung der Erfüllbarkeit von Qualitätsattributen zur Entwurfszeit.

Ein praxisrelevantes Beispiel für eine solche Sprache ist die AADL [FGH06] aus der Automotive-Domäne. Die „Architecture Analysis and Design Language“ vereint die Modellierung automotiver Systemarchitekturen mit der Annotation von physikalischen Eigenschaften über das System. So werden neben funktionalen auch qualitative Anforderungen frühzeitig überprüfbar.

Palladio Komponentenmodell

In dieser Arbeit wird das Palladio Komponentenmodell [RBB+11] (Englisch: PCM, Palladio Component Model) als Architekturbeschreibungssprache (Englisch: Architecture Description Language, ADL) zur Spezifikation der Architektur und zusätzlicher Modellartefakte zur qualitativen Analyse genutzt.

Der Palladio-Ansatz [RBH+16] unterstützt den Anwender in frühen Entwicklungsphasen in der Abwägung von Entwurfsentscheidungen. Nach der Modellierung einer komponentenorientierten Softwarearchitektur und Infrastrukturen zu deren Deployment, sind frühzeitige Beurteilungen der Systemqualität möglich. Dafür werden pro Softwarekomponente rudimentärer Verhaltensbeschreibungen (SEFF, Englisch: Service Effect Specification) modelliert, die festlegen in welcher Intensität Ressourcen der Infrastruktur genutzt werden.

Zur Beurteilung von Entwurfsentscheidungen existieren unterschiedliche Analyseansätze auf Basis des PCM. Der Kernansatz ermöglicht die architekturbasierte Beurteilung von Performanz, Zuverlässigkeit, Wartbarkeit und Kosten eines Systementwurfs, [BKR09]. Durch die architekturzentrierte Ausrichtung, wird die wiederholte Implementierungen von Systemanpassungen für Simulationen (SIL, Software in the Loop) oder Ausführungen auf Teststänlden (HIL, Hardware in the Loop) verringert.

Eine Übertragung der Ergebnisse PCM-basierter Analysen in die Betriebsphase eines Systems ist ebenfalls möglich. So realisiert Matevska et al. [MH07, Mat09] das Re-Deployment von Softwarekomponenten zur Laufzeit unter Bewahrung der Verfügbarkeit und Dienstgüte des Systems. Die transaktionale Durchführung von Rekonfigurationen stützt sich dabei auf fundamentale Konzepte von Kramer und Magee [KM85, KM90].

In dem vorliegenden Ansatz wird ein PCM-basiertes Systemmodell um zusätzliche technischen Spezifikationen aus einer Ressourcenplattform ergänzt. Dieses Wissen wird zur Nutzwerterhebung und strukturellen Abgrenzung von Ausprägungen der Softwarearchitektur aufgegriffen. Mögliche Ausprägungen werden dabei unter Einsatz einer variantenreichen Modellierung automatisiert abgeleitet.

2.2.2. Variantenreicher Systementwurf

Die Beschreibung und Analyse der Variabilität im Entwurfsraum ist Gegenstand der Produktlinienforschung. In dieser Arbeit werden unterschiedliche Techniken zur Variabilitätsbeschreibung genutzt.

Feature-orientierte Domänenanalyse

Die Feature-orientierte Domänenanalyse (FODA) [KCH+90] ist ein stark verbreiteter Ansatz zur explizite Variabilitätsbeschreibung eines Systems. Laut Kang et al. beschreiben ein *Feature* als markantes oder sichtbares Merkmal, Qualität oder Eigenschaft eines (Software)systems. Ein Variabilitätsmodell beschreibt Gemeinsamkeiten und Unterschiede und wird als *Problemraum* bezeichnet. Entwurfsmodelle, die entlang des Variabilitätsmodells ausgeprägt werden, bilden den *Lösungsraum*. Kang et al. [KCH+90] führen zur Notation des Problemraums Feature-Modelle ein. Diese Modelle beschreiben Beziehungen zwischen Features in einer hierarchische Baumstruktur. Benavides et al. [BSRC10] erläutern die Elemente eines Feature-Modells entlang des Beispiels in Abbildung 2.1.

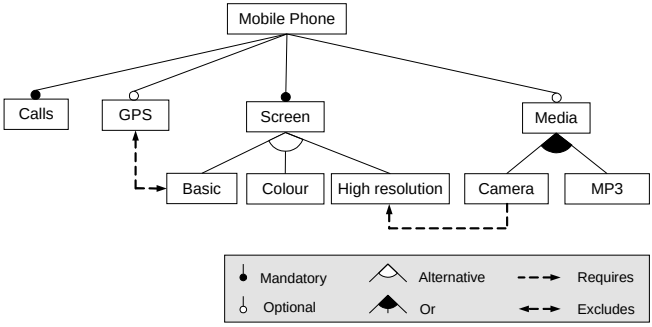


Abbildung 2.1.: Beispiel eines Feature-Modells (Quelle: [BSRC10])

Geforderte (Englisch: Mandatory) Features treten in einem Produkt auf, sobald das Eltern-Feature im Baum ausgewählt wurde. Im Beispiel muss jedes Mobiltelefone zum Beispiel Anrufe ermöglichen. Die nicht-verpflichtende Auswahl eines Features wird als optional (Englisch: Optional) bezeichnet. Ein Telefon kann optional ein GPS-Modul enthalten. Befindet sich ein Feature in einer Alternativ-Gruppe (Englisch: Alternative) kann immer nur wechselseitig eines der Geschwister ausgewählt werden. In dem Beispiel kann immer nur eine Art von Display gewählt werden. Eine Oder-Gruppe (Englisch: Or) ermöglicht hingegen die Auswahl von einem bis allen benachbarten Features. Ein Handy kann eine Kamera- und MP3-Funktion enthalten. Durch die

Optionalität des Media-Features kann jedoch auch beides entfallen. Weiterhin werden Querbeziehungen als zusätzliche Nebenbedienungen zwischen Features in unterschiedlichen Teilbäumen erfasst. So beschreiben Ein- und Ausschlussbeziehungen (Englisch: Require beziehungsweise Exclude) ob ein Feature ein anderes voraussetzt oder hingegen ausschließt. In dem Beispiel bedingt eine Kamera auch den Einbau eines hochauflösenden Displays. Ein GPS-Modul hingegen ist nicht mit einem Basis-Display kompatibel.

Liegt eine gültige Belegung von Features (*Feature-Konfiguration*) über alle Bedingungen im Feature-Modell vor, ist ein Produkt (*Variante*) aus dem Lösungsraum valide.

Modellierung variantenreicher Lösungsräume

Schaefer et al. [SRC+12] fassen den Stand der Technik und mögliche Perspektiven im Bereich der Softwareproduktlinien zusammen. Mit Bezug auf die Bildung des Lösungsraums, wird auf zwei vorrangig eingesetzte Techniken [VG07] zur Modellierung verwiesen. Dabei besteht vor allem bei dem Umfang des Ausgangsmodells ein wesentlicher Unterschied.

Annotative Ansätze oder überlagernde Varianten [CA05] arbeiten nach dem Prinzip der Subtraktion (Englisch: Negative Variability) von Bestandteilen eines Systemmodells, welches alle Varianten der Produktlinie vereint. Im Modell entstehen dabei Widersprüche und Inkonsistenzen. Erst durch die Annotation von Elementen eines korrespondierenden Variabilitätsmodells (zum Beispiel Feature Model) ist die Ableitung von Produkten möglich.

Entgegen arbeiten *kompositionale Ansätzen* additiv (Englisch: Positive Variability). Ausgehend von einem leeren Modell, werden neue Modellelemente entlang der Restriktionen eines Variabilitätsmodell zusammengesetzt.

Neben den genannten Ansätzen stellt die *Delta-Modellierung* [CHS11] die Variabilität durch modulare Transformationen auf Modellartefakten dar. Mengen von Transformationen (Deltas) beschreiben die Bildung von Produkten aus einem gemeinsamen Kernmodell. Ausgehend von diesem Kern mit wenigen bis keinen Modellelementen, werden neue Modellelemente hinzugefügt, modifiziert oder auch entfernt. Die Elemente spezifizieren zusätzliche Funktionalitäten zur Erfüllung der Features in einem Variabilitätsmodell. Zusätzlich zu diesen Transformationen ausgehend vom Kernmodell, sind

Regressionsdeltas zwischen einzelnen Varianten ableitbar. Die Delta-Modellierung wurde anfangs auf Programmiersprachen [SBB+10] angewandt, aktuelle Arbeiten übertragen das Prinzip unter anderem auf Softwarearchitekturen [HRRS14] und Zustandsmaschinen [LLL+14].

In dieser Arbeit erfolgt zunächst eine Feature-orientierte Analyse des Problemraums. Eine Überführung der Erkenntnisse in eine Produktlinie, modelliert als Feature Modell, erfolgt dabei im Zuge des Domain Engineerings [PBDL05].

Freiheitsgrade für komponentenorientierte Systeme

Zur Aufspannung des Lösungsraums folgt die vorliegende Arbeit einer zur Delta-Modellierung ähnlichen Methode. Analog zu frühen Ansätzen der komponentenorientierte Produktlinien, vergleiche [ABB+02], werden die Entwurfsmodelle unmittelbar um Variationen angereichert. Diese Arbeit stützt sich dabei auf das Konzept der *Freiheitsgrade* [KR11, Koz11] (Englisch: Degrees of Freedom) für Komponentenmodelle. Der Ansatz definiert eine umfangreiche Auswahl an Variationen innerhalb komponentenorientierter Modelle und erlaubt im weiteren Verlauf die automatische Optimierung der Architekturauswahl.

Der Ansatz in dieser Arbeit greift ausschließlich auf den Freiheitsgrad zur *Komponentenselektion* zurück. Unterschiedliche Optionen für eine Komponentenausprägung werden zunächst in einer Palladio-Architektur modelliert und als alternative Optionen in einem Freiheitsgrad für eine spezifische Komponente aufgenommen. Die Menge aller Freiheitsgrade definiert den *Entscheidungsraum* gültiger Architekturausprägungen.

2.2.3. Multikriterielle Optimierung des Entwurfsraums

Auf Grundlage eines einheitlichen Bewertungsmaßstabs für Lösungskandidaten (Optimierungsziel), ermöglichen Optimierungsansätze die Identifikation adäquater Lösungen in Polynomialzeit. Ist eine Vielzahl von Optimierungszielen von jedem Kandidaten zu erfüllen, handelt es sich um ein multikriterielles Optimierungsproblem.

Pareto-Effizienz

In dieser Klasse von Optimierungen, treten oftmals unvergleichliche oder konkurrierende Optimierungsziele (qualitative Eigenschaften) auf. In einem Hardwaresysteme stehen sich beispielsweise Energieverbrauch und Leistung im Allgemeinen gegenüber. Anstelle globaler Optima, werden mögliche Lösungen in Mengen von *Pareto*-effizienter Kandidaten gebildet.

Die Pareto-Optimalität beschreibt dabei den Zustand, dass eine Eigenschaft eines Kandidaten nicht bessergestellt werden kann, ohne eine andere Eigenschaft des gleichen Kandidaten zu verschlechtern. Pareto-effizienten Kandidaten bilden eine Hyperebene im Raum der betrachteten Eigenschaften. Je nach Richtung der Optimierung (Minimierung beziehungsweise Maximierung) bildet diese Ebene die Untergrenze beziehungsweise Obergrenze aller untersuchten Kandidaten im Entwurfsraum. Kandidaten die oberhalb beziehungsweise unterhalb dieser *Pareto-Front* liegen sind *Pareto-dominiert*.

Meta-heuristische Kandidatenexploration

Zur Identifikation von Pareto-effizienter Kandidaten innerhalb eines Entwurfsraums, sind grundsätzlich sämtliche Kandidaten zu analysieren. Die Erzeugung aller Ausprägungen führt jedoch zu einem kombinatorischen Rechenaufwand. Zur multi-kriteriellen Optimierung wird daher in dieser Arbeit ein meta-heuristisches Suchverfahren anstelle einer erschöpfenden Suche eingesetzt. Jedes Kriterium wird dabei durch eine qualitative Eigenschaft im mehrdimensionalen Optimierungsraum (*Qualitätsdimension*) repräsentiert.

In dieser Arbeit wird ein genetischer Algorithmus, vergleiche [Hol92], zur Lösung des Optimierungsproblems genutzt. Dies bietet sich an, da so eine hohe Streuung über alle Qualitätsdimensionen schnell durchsucht werden kann. Den hohen Berechnungsaufwänden wirken relativ kostengünstige Validierungen und Wertevaluierungen pro Kandidat in dieser Arbeit entgegen. Der gewählte Algorithmus *NSGA-II*[2] [DPAM02] liefert bei minimalem Parametrisierungsaufwand eine hohe Performance. Die Abbildung 2.2 zeigt das grundlegende Prinzip eines genetischen Algorithmus.

[2] Nondominated Sorting Genetic Algorithm II

Der Anwender oder ein Generatoralgorithmus erzeugt zunächst eine initiale Population von Kandidaten. Sämtliche Kandidaten in der Population werden unter Verwendung der qualitativen Eigenschaften und einer Zielfunktion kriterienweise bewertet. Im Anschluss wird eine Iteration über genetische Operationen gestartet. Zunächst werden aus der vorliegenden Population valide Kandidaten als Eltern selektiert. Im Anschluss werden die Eltern rekombiniert (Englisch: cross-over), um zwei Kinder zu erzeugen. Zur Steigerung der Diversität werden einzelne Gene der Kinder mutiert. Aus den Kindern entsteht eine neue Population und der Zyklus beginnt erneut. Optional können invalide Kinder auch durch zusätzliche Operatoren repariert werden. Dies wird in dem Ansatz dieser Arbeit jedoch nicht behandelt. Das Verfahren terminiert, sobald ein anwenderdefiniertes Abbruchkriterium erfüllt ist. Mögliche Abbruchkriterien sind zum Beispiel eine feste Anzahl von Iterationen oder auch das Erreichen einer Mindestgröße der Pareto-effizienten Menge. In dieser Arbeit wird die erstgenannte Strategie verfolgt.

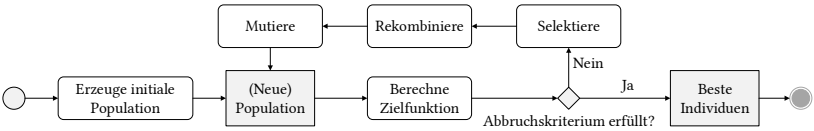


Abbildung 2.2.: Grundprinzip genetischer Algorithmen

Nach Ausführung des Algorithmus liegt eine Menge Pareto-effizienter Kandidaten vor und eine weitere Menge der dominierten Kandidaten. Da es sich um eine Metaheuristik handelt, ist nicht garantiert, dass der gesamte Entwurfsraum traversiert wurde. Entsprechend können Kandidaten aus beiden Kategorien unentdeckt bleiben.

Zur Übertragung dieses Prinzips zur Erzeugung eines Rekonfigurationsraums wird in dieser Arbeit die Validität eines erzeugten Kandidatens auf Grundlage der definierten Freiheitsgrade, vergleiche Abschnitt 2.2.2, überprüft. Dabei wird auf eine existierende Implementierung im Palladio-Umfeld (PerOpteryx [MKBR10]) aufgegriffen und erweitert. Die Bewertung erfolgt entlang der annotierten technischen Spezifikationen (als Qualitätsattribute), vergleiche Abschnitt 2.2.1, für jede Komponente des Kandidatens.

Zwei verwandte Ansätze optimieren die Architekturauswahl ebenfalls entlang evolutionärer Algorithmen. Der Ansatz von Li et al. [LEEC11] evaluiert unterschiedliche Algorithmen zur Optimierung einzelner Architekturen. Aleti et al. [ABGM09] optimieren AADL entlang eines Optimierungskriteriums.

2.2.4. Architekturevaluation

Die Lösungsmengen aus multi-kriteriellen Optimierungen enthalten per Konstruktion Kandidaten mit individuellen qualitativen Eigenschaftswerten. Im Zuge der Architekturevaluation sind die Kandidaten anhand kumulierter Nutzwerte voneinander abzugrenzen.

Szenariobasierte Ansätze

Dobrica et al. [DN02] fassen gängigen Techniken zur Architekturevaluation zusammen. Darunter sind auch zwei etablierte szenariobasierte Bewertungsverfahren, die für diese Arbeit relevant sind.

Die Zielsetzungen des Anwenders an das System stehen im Mittelpunkt einer szenariobasierten Architekturbewertung. So werden Szenarien definiert und priorisiert, die entlang einer Architektur qualitativ ausgewertet werden. Das Verfahren wurde erstmalig durch Kazman et al. [KABC96] unter dem Namen *SAAM* (Englisch: Software Architecture Analysis Method) eingeführt. SAAM adressiert hauptsächlich die Verbesserung der Wartbarkeit einer Architektur. Dazu werden aus Befragungen mit Prozessbeteiligten Szenarien abgeleitet und kritische Elemente innerhalb der Architektur begutachtet. Dieses Vorgehen forciert einerseits einen hochqualitativen Architekturentwurf und andererseits eine ausführliche Dokumentation in der Entwicklung.

Analog zu den Herausforderungen in der multi-kriteriellen Optimierung, vergleiche Abschnitt 2.2.3, treten auch in der Architekturbewertung Zielkonflikte (Englisch: Trade-Off) und die Unvergleichbarkeit zwischen den Bewertungskriterien auf. *ATAM* [KKC00] (Englisch: Architecture Tradeoff Analysis Method) adressiert diese Herausforderung in frühen Entwicklungsphasen. Das Verfahren analysiert Folgen in Bezug auf geplante Änderungen

im Architekturentwurf. Dazu werden umfangreiche Entwurfsentscheidungen in einem aufwendigen Prozess unter Mitwirkung aller Prozessbeteiligten evaluiert. Im Gegensatz zu SAAM steht hier nicht die Wartbarkeit im Vordergrund, sondern die Identifikation und Minimierung von Risiken durch falsche Entwurfsentscheidungen.

Grunske [Gru07] integriert eine Vielzahl von Bewertungsmethoden in einen generischen Ansatz zum Wissenstransfer zwischen unterschiedlichen Qualitätsdomänen von frühen Entwurfsphase bis hin zu Simulationen.

Zur ganzheitlichen Bewertung der Kandidatenmengen aus der multi-kriteriellen Optimierung, wird in dieser Arbeit ein Ansatz zur Nutzwertbildung aufgegriffen und für variantenreiche komponentenorientierte Systeme erweitert.

Nutzwertbildung zur Architekturevaluation

Florentz et al. [FH06] bieten einen integrierten Ansatz zur Resolution von Zielkonflikten im Rahmen der Architekturevaluation für eingebettete Systeme an. Jedes Kriterium wird in eine Evaluationsstruktur eingebunden und trägt in einer hierarchische Auswertung zu einem Teil eines ganzheitlichen Nutzwertes bei.

Diese Qualitätsattributsgraphen (QADAG, Quality Attribute Directed Acyclic Graph) sind baumartig aufgebaut und verweisen in den Knoten auf die Qualitätsattribute, die zu bewerten sind. Diese Qualitätsattribute greifen Bewertungsszenarien (vergleiche SAAM) auf und sind mit typisierten Werten belegt. Die Werte können durch Bedingungen limitiert und zur Weiterverarbeitung interpretiert werden. Entsprechend dem Grundproblem der multi-kriteriellen Optimierung, sind die Werte der Qualitätsattribute von unterschieden Datentypen und nicht ineinander überführbar. Eine Interpretation bildet jeden Wert eines Qualitätsattributs auf eine prozentuale Skala ab. Florentz et al. führen dafür Qualitätsraten als Nutzwertfunktionen ein, siehe Abbildung 2.3.

Zur Aufstellung der Funktion ist auf Expertenwissen aus der Fachdomäne zurückzugreifen, um einen absoluten Wert auf einen Nutzwert des Attributs abzubilden. Ausgehend von der x-Achse, wird die Qualität von der y-Achse abgelesen. In der Abbildung bildet so die Last auf einem Datenbus von 39%

auf eine Qualität von 90% ab. Dieser interpretierte Qualitätswert kann im Anschluss mit weiteren Kriterien, zum Beispiel der Prozessorauslastung, in Verhältnis gesetzt werden.

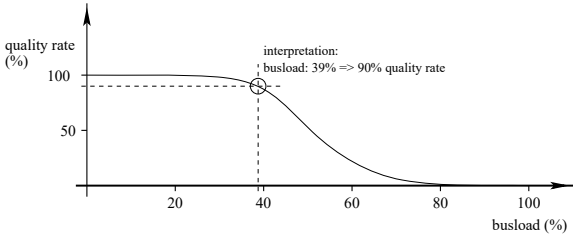


Abbildung 2.3.: Qualitätsrate zur Nutzwertbildung (Quelle: [FH06])

Die Struktur eines QADAG wird durch ein Metamodell definiert, welches Qualitätsattribute in den Mittelpunkt der Betrachtung stellt. Die Abbildung 2.4 zeigt das Metamodell.

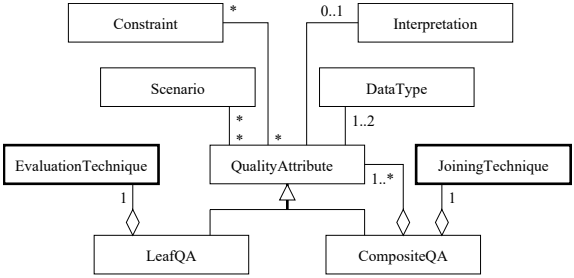


Abbildung 2.4.: QADAG-Metamodell (Quelle: [FH06])

Die Blätter (LeafQA) erben die Eigenschaften eines Qualitätsattributs und nutzen eine Evaluationstechnik (EvaluationTechnique). Diese Technik beschreibt, wie die Werteerhebung realisiert ist, zum Beispiel als Konstante, mit einer Code- oder Modellmetrik. Zusammengesetzte Knoten (CompositeQA) treten in höheren Ebenen des Baums auf. Knoten verweisen auf weitere innere Knoten oder Blätter. Eine Verknüpfungsstechnik (JoiningTechnique) beschreibt dabei eine Funktion zur Kumulation der Werte in den Knoten.

Eine naheliegende Umsetzung einer Funktion ist die gewichtete Summe. Die Wurzel des QADAG ist ebenfalls ein zusammengesetzter Knoten. Durch die Auswertung der Evaluationstechniken und der Verrechnung der einzelnen Knotenwerte in der Kumulation entsteht als Gesamtergebnis eine ganzheitliche Bewertung der betrachteten Architektur, der *Nutzwert der Architektur*.

Der Ansatz von Florentz et al. ermöglicht die strukturierte und transparente Kombination verschiedenartigen Qualitäten und eine Auflösung von Trade-Offs durch das Festlegen konkreter Verhältnisse. Die Autoren beschreiben weiterhin wie Sensitivität in den Trade-Offs identifiziert und Schwellwerte optimiert werden. Im Fokus dieser Arbeit liegt die Verwendung von Qualitätsattributsgraphen zur repräsentativen Kodierung qualitativer Aspekte von Benutzungsszenarien eines Software-/Hardwaresystems.

2.2.5. Selbstmanagement

Das Selbstmanagement in technischen Systemen ist natürlichen Prozessen nachempfunden, die unbewusst von einem Organismus gesteuert werden [KC03]. Beispiele dafür sind der menschliche Herzschlag oder die Atmung. Eine kontinuierliche Anpassung der Prozesse erfolgt autonom.

Kategorien des Selbstmanagements

Ganek und Corbi, [GC03] verfeinern den Begriff des Selbstmanagements in vier Kategorien. Die *Selbstkonfiguration* beschreibt die Anpassung des Systems an veränderte Ausführungskontexte durch autonome Rekonfiguration. Die Detektion, Diagnose und Behandlung von Fehlern wird durch die *Selbstheilung* realisiert. Der Prozess der *Selbstoptimierung* überwacht Ressourcen und passt diese fortlaufend an die Betriebs- und Kundenanforderungen an. Die *Selbstabsicherung* sagt Bedrohungen für das System voraus, identifiziert und vermeidet diese.

Unter Beachtung dieser Selbst-Eigenschaften (Englisch: Self Properties) passt sich ein autonomes System an neue Betriebskontexte an, regiert auf Änderungen, Fehler oder Bedrohungen und optimieren sich entsprechend der Anforderungen.

Ausgehend von einer neuen Zielsetzung, beispielsweise eine Änderungsanfrage oder ein Fehlerfall, wird ein Kreislauf zur Anpassung des System initiiert. Die Abbildung 2.5 zeigt die Zusammenhänge in einem geschlossenen Kreislauf.

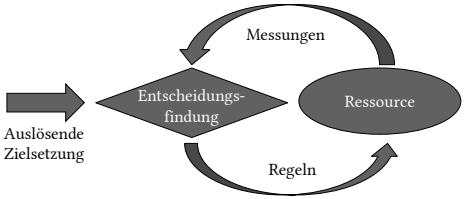


Abbildung 2.5.: Kreislauf der Entscheidungsfindung (Quelle: [GKKM10], angepasst)

Ausgehend von der Identifikation und Festlegung einer Entwurfs- beziehungsweise Rekonfigurationsentscheidung, modifizieren Regeln eine Ressource (Software oder Hardware) des Systems. Im Anschluss erfolgen Messungen, um den Erfolg der Änderungen zu beurteilen. Sofern notwendig, folgen weitere Iterationen zur Optimierung. Änderungen werden von außerhalb angestoßen, jedoch ohne weitere Zusatzinformationen selbstständig vom System verarbeitet.

Selbstmanagement nach MAPE-K

Kephart und Chess [KC03] erweitern den Kreislauf des Selbstmanagement und grenzen die Prozessschritte voneinander ab. Ein *Autonomie-Manager* verwaltet ein Element, welches eine Ressource oder Subsystem repräsentiert. Innerhalb des Managers liegt der *MAPE-K*-Kreislauf zur Überwachung, Analyse, Planung und Ausführung unter Verwendung einer Wissensbasis (Englisch: Monitoring, Analyse, Plan, Execute, Knowledge) vor. Der erweiterte Steuer-/Regelkreis, dargestellt in Abbildung 2.6, führt dabei die folgenden Schritte sequentiell aus.

An definierten Messpunkte sammelt ein *Monitor* Daten über das *verwaltete Element*. Der *Analyse*-Schritt interpretiert die erhobenen Werte und leitet den Systemzustand daraus ab. Treten Abweichungen auf, werden Aktionen zur Wiederherstellung ausgewählt. Der *Planer* wertet die Aktionen

auf Anwendbarkeit und Reihenfolge bezüglich der Betriebsanforderungen aus. Abschließend stößt *Execute* die Ausführung der vorsortierten Aktionen im verwalteten Element durch definierte Regeln an. Sämtliche Schritte nutzen eine gemeinsame Wissensbasis (*Knowledge*) zur Ablegung und Auswertung von Messdaten und zur Entscheidungsfindung auf Grundlage von vordefinierten Modifikatoren. Das Wissen über das System repräsentiert den beobachtbaren Systemzustand, zum Beispiel in Protokolldaten (Englisch: Log Data).

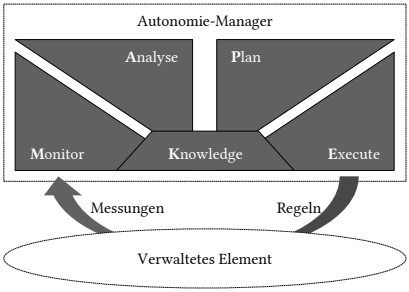


Abbildung 2.6.: MAPE-K Regelkreis (Quelle: [KC03], angepasst)

In dieser Arbeit wird das Selbstmanagement eines fehlertolerantes Systems zur Abgrenzung der Aktivitäten im Rekongurationsprozess adressiert. Weiterhin wird die Rolle eines zentralen Modelles als Wissensbasis zur Wartungsunterstützung eingeordnet. Der Grad an Autonomie wird in dem anvisierten Ansatz ausdrücklich auf die Planung von Wartungsaktivitäten reduziert. Die erzeugten Regeln modifizieren das verwaltete System somit nicht unmittelbar. Eine Beteiligung der Entscheidungsträger ist anvisiert; eine adaptive und autonome Modifikation wird nicht betrachtet.

2.2.6. Empirische Studien

Im Rahmen der Validierung des Ansatzes wird eine empirische Studie unternommen. Die Studie wird in Form von Interviews durchgeführt. Dies erfordert die Ableitung eines Fragebogens aus den Zielsetzungen dieser Arbeit. Ein Überblick über die dafür relevanten Techniken folgt.

Operationalisierung

Im Zuge der *Operationalisierung* werden die Forschungsfragen des Ansatzes für den *kulturellen Kontext* [GL10] des befragten Experten aufbereitet. Dies eliminiert Verständnisprobleme und ermöglicht eine effektive Beantwortung. Nach Kaiser [Kai14] unterteilt sich die Operationalisierung in zwei Phasen, siehe Abbildung 2.7. Zunächst werden *Analysedimensionen* und *Fragenkomplexe* erhoben, anschließend werden konkrete *Interviewfragen* aufgestellt.

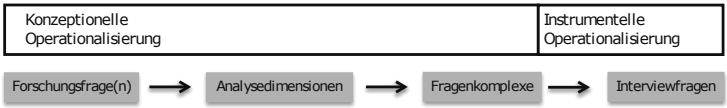


Abbildung 2.7.: Phasen der konzeptionellen und instrumentellen Operationalisierung im Interviewentwurf (Quelle: [Kai14])

Die Ableitung einer Analysedimension richtet sich stark nach den Erkenntniszielsetzungen des Interviews. Die Dimensionen extrahieren systematisch die relevanten Kernpunkte aus den Forschungsfragen. Die Fragenkomplexe strukturieren hingegen die Inhalte empirisch messbar in zusammenhängende Kategorien.

Als erste Phase der Operationalisierung werden zunächst die detaillierten Forschungsfragen in Teilaspekte in Form der Analysedimensionen heruntergebrochen, die eine Beobachtung der Phänomene ermöglichen. Die Dimensionen besitzen einen maßgeblichen Bezug zu den Begriffen aus der Grundlagenliteratur des betrachtenden Forschungsfelds, beziehungsweise greifen Methodiken des Arbeitsumfelds und vorhandene Begriffe auf. Ein Interview dient zur Beurteilung des Vorhandenseins und der Ausprägung der antizipierten Phänomene.

Als nächsten Schritt der konzeptionellen Operationalisierung verfeinern Fragenkomplexe die Analysedimensionen. Kaiser fasst diesen Schritt mit der Festlegung der Kriterien zur Beobachtung und Erhebung der Analysedimensionen zusammen. Fragenkomplexe vertiefen bestehende Festlegungen aus dem beobachteten Forschungs-/Anwendungsfeld und verbinden diese mit der angestellten Untersuchung. Ein wesentlicher Fokus in der Definition der Fragenkomplexe liegt auf der Herstellung eines thematischen Bezugs der

Expertise der befragten Personen zu den Analysedimensionen. Die Fragenkomplexe müssen für den Experten verständlich ausformuliert sein, jedoch ausreichend Raum für die anvisierten Erkenntnisinteressen lassen. In der Auswertung dienen die Fragenkomplexe der Einordnung und Relevanz der Aussagen der befragten Experten.

Im Übergang von der konzeptionellen zur instrumentellen Operationalisierung werden die Fragenkomplexe in konkrete Interviewfragen transformiert. Laut Kaiser liegen die Schwerpunkte in dieser Phase vorrangig in der Transformation der abstrakten Definitionen in greifbare Fragen. Zusätzlich wird in dieser Phase über Reihenfolge, Detaillierungsgrad und Fragetypen entschieden.

Die Rückführung der Erkenntnisse erfolgt in entgegengesetzter Prozessreihenfolge. Dabei ist das wesentliche Ziel, die erhobenen Aussagen zur Beantwortung der Forschungsfragen zu priorisieren, zu bündeln und sachgerecht zu abstrahieren.

Fragetypen

Fragen treten nach Kvale [Kva96] sowie Stigler und Felbinger [SF05] hauptsächlich in folgenden Typen in einem Interview auf. Zu Beginn der Befragung werden *Einführungsfragen* verwendet, um den Experten in seinem Wissensgebiet abzuholen. *Strukturierende Fragen* werden dazu eingesetzt unterschiedliche Themengebiete einzuleiten. Eine *Direkte Frage* wird benutzt, um fixe Kernaussagen der Experten zu erheben. Der Einsatz dieser Technik sollte defensiv eingesetzt werden, um den Verlauf des Interviews für den Befragten angenehm zu gestalten. Allgemeine Aussagen, Einschätzungen und Positionen werden in *indirekten Fragen* erfasst. Dieser Fragetyp tritt häufig auf und lässt ein großes Maß an Freiheit in der Beantwortung. Tiefgehenden Details zu einem Sachverhalt werden durch eine *spezifizierende Frage* erhoben. Diese Technik ist im Laufe des Interviews spontan anzuwenden, wenn weiterer Klärungsbedarf zu einer vorherigen Aussage besteht. Teilweise treten entsprechende Fragestellungen erst nach dem Abschluss des Interviews auf und sind möglicherweise in einer Folgebefragung zu bearbeiten. Der Fragetyp *interpretierende Frage* ist dann anzuwenden, wenn der Befragte bereit ist seine persönlichen Interpretationen preiszugeben.

Während der Auswertung helfen die Antworten bei der Einordnung der weiteren Aussagen im Laufe des Interviews.

Experteninterview

Ein strukturiertes Interview unterstützt die wissenschaftliche Erhebung und Einordnung von Domänenwissen. Das Format dieser Datenerhebung kann unterschiedliche Formen annehmen. Neben den Typen Ethnographisches Interview und Narratives Interview, ist das *Experteninterview* auf die Fallstudien-bezogene Erkenntnisgewinnung ausgelegt [Kai14].

Die Auswahl der Experten hat einen wesentlichen Einfluss auf die Relevanz der erhobenen Daten. Gläser und Laudel [GL10] weisen dabei auf drei Kernaspekte hin. Zunächst ist sicherzustellen, dass der Experte Zugriff auf die relevanten Information hat. Dies beinhaltet auch, ob der Experte eine Einordnung zwischen den Fragen und den verfügbaren Informationen herstellen kann. Darauf aufbauend ist zu klären, welcher Experte diese Informationen präzise wiedergeben und mit zusätzlichen Angaben ausgestalten kann. Letztlich sind Experten auszuwählen, die ebenfalls bereit und ermächtigt sind die Informationen im Interview zu verwenden beziehungsweise diese auf eine entsprechende Abstraktionsebene anzuheben, um zum Beispiel eine Geheimhaltung von Detaildaten einzuhalten.

Um die fachlichen Inhalte der Befragung vorab zu definieren und die Asymmetrie im Kenntnisstand zu den Fachgebieten des Experten und des Fragenden auszugleichen, wird die Befragung strukturiert durchgeführt. Ein *Interviewleitfaden* dient dazu die Befragung des Fachpersonals zielführend vorzubereiten und kontrolliert durchzuführen.

Qualitative Inhaltsanalyse

Nachdem eine Befragung durchgeführt wurde, können für Experteninterviews eine Vielzahl möglicher Ansätze zur qualitativen Inhaltsanalyse angewandt werden, vergleiche [May13] oder [Pat90]. Kaiser [Kai14] fasst die Analyse in drei Schritten zusammen. Zunächst werden die Antworten in geeigneter Form in Textform überführt, dann werden die Kernaussagen anhand der Analysedimensionen identifiziert und letztlich wird ein Bezug zwischen

den Aussagen und untersuchten Aspekten erzeugt, um eine Interpretation zu ermöglichen.

Zur Erfassung der Antworten in Textform ist eine Transkribierung grundsätzlich notwendig. Kaiser stellt dabei heraus, dass aufgrund des hohen zeitlichen und dokumentarischen Aufwands zumeist nur relevante Passagen als Text erfasst werden. Dabei ist es zielführend eine Paraphrasierung in eigenen Worten anstelle von wörtlichen Zitaten entlang einiger Grundregeln zu nutzen, wobei Wiederholungen und ausschmückende Formulierungen unterdrückt werden sollten. Eine Umsetzung sollte allerdings weiterhin entsprechend des Interviewverlaufs chronologisch durchgeführt werden. Fragen und Antworten müssen deutlich gekennzeichnet werden. Unverständliche Passagen sind entsprechend zu annotieren. Das Transkript sollte auf geeignete Art indiziert werden, um Querverweise zu ermöglichen.

3. Prozess

Der Ansatz beschreibt einen dreiphasigen Prozess zur Entscheidungsunterstützung für Wartungsaufgaben, der sich von Vorausberechnungen zur Entwurfszeit bis hin zur Fehlerbehandlung im Betrieb erstreckt.

3.1. Übersicht

Wie in Abschnitt 1.2.2 motiviert, beginnt der Prozess bereits frühzeitig im Entwurf. Eine systematische Voruntersuchung des Entwurfsraums hinsichtlich Validität und Austauschbarkeit alternativer Systementwürfe bildet die Grundlage einer effektiven und zugleich effizienten Entscheidungsunterstützung im Betrieb. So wird bereits vor der Inbetriebnahme des Systems Wissen über mögliche Rekonfigurationen und deren Beziehungen akquiriert, vorausberechnet und aufbereitet.

Der in dieser Arbeit entwickelte Prozess unterteilt sich in die *Entwurfsraumerzeugung*, die *Architekturrelationsbildung* sowie die *Alternativenexploration*, siehe Abbildung 3.1. Zunächst wird der Entwurfsraum aufgespannt und optimiert, anschließend wird dieser entlang eines Betriebsszenarios kostenorientiert priorisiert und zuletzt wird eine Rekonfiguration in Abhängigkeit eines Fehlers angestoßen und dokumentiert. Die Abbildung 3.1 zeigt den Prozess als Aktivitätsdiagramm mit allen Eingabemodellen, erzeugten Artefakten sowie automatisierten Transformationen und anwendergesteuerten Aktionen.

Die folgenden Abschnitte erläutern die einzelnen Prozessschritte im Detail. Zur Veranschaulichung werden wesentliche Aspekte unter Verwendung des durchgängigen Beispiels aus Abschnitt 1.5 aufbereitet. Dabei wird insbesondere auf die Umsetzung einer fehlertoleranten Umfelderfassung (Tracking),

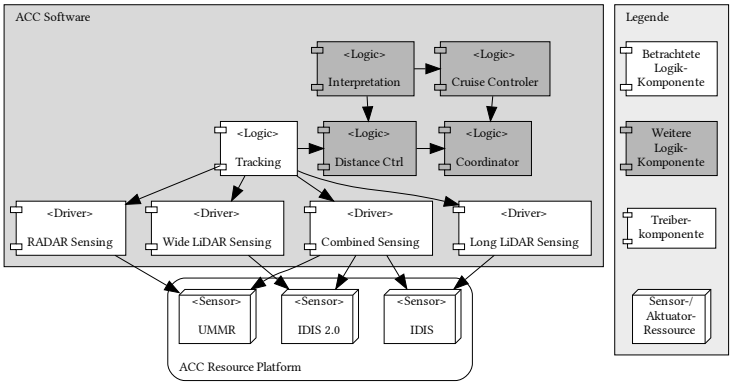
unter Verwendung der vorab eingeführten redundanten Sensorik, eingegangen.

3.2. Entwurfsraumerzeugung

Zur Entwurfszeit werden aus einem Systemmodell, vergleiche Abschnitt 4.1.1, in Verbindung mit einem Variabilitätsmodell eine Menge von Architekturkandidaten entlang definierter Variationspunkte abgeleitet, vergleiche Abschnitt 4.2.1 und Abschnitt 4.2.2.

Durchgängiges Beispiel 3.1 zeigt eine initiale Softwarearchitektur mit Variationspunkten und Ressourcenabhängigkeiten. Die Architektur fokussiert sich hier auf die Umsetzung der Tracking-Komponente innerhalb des ACCs aus Abschnitt 1.5.4.

Durchgängiges Beispiel 3.1: Initiale Softwarearchitektur



Die ACC-Software basiert auf diversen Komponenten zur Datenverarbeitung von Fahreraktionen, Werten aus dem Fahrzeugsystem und insbesondere Sensorwerten. Die Sensoren werden im Komponentenmodell als Treiberkomponenten dargestellt, die von dem Zugriff auf Ressourcen abstrahieren. Die Treiberkomponenten sind optional zueinander auswählbar und daher gestrichelt dargestellt.

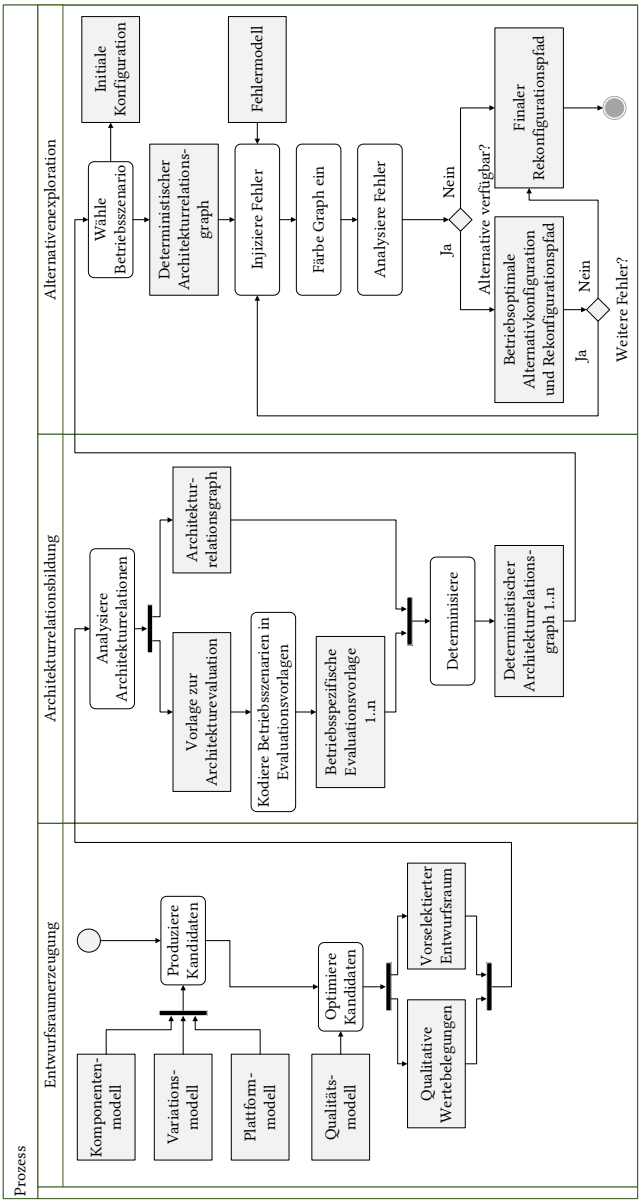
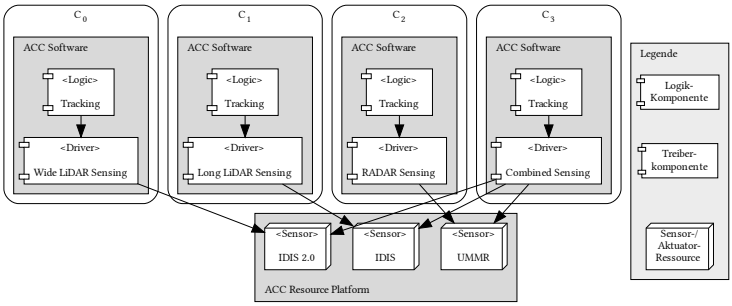


Abbildung 3.1.: Prozess zur Entscheidungsunterstützung für Wartungsaufgaben

Zur Berücksichtigung von Bedingungen, die sich auf Ebene der Hardwareressourcen ergeben, werden zusätzliche Informationen über Ressourcenbeziehungen in der Kandidatenproduktion aufgegriffen. Dabei werden pro Kandidat sämtliche Ein- und Ausschlussbeziehungen genutzter Sensorik und Aktuatorik auf Gültigkeit überprüft. Die Menge der produzierten Kandidaten wird iterativ gesteigert und entlang gegebener Qualitätsdimensionen mit Prognosedaten aus Datenblättern bewertet. Sobald ein gefordertes Maß an Optimalität erreicht ist, resultiert als *Entwurfsraum* eine Menge *Pareto-optimaler Konfigurationen*. Eine Konfiguration wird charakterisiert durch eine Softwarearchitektur mit spezifizierten Anforderungen an Elemente aus der Gesamtmenge von Hardwareressourcen.

In Durchgängiges Beispiel 3.2 resultieren vier mögliche Konfigurationen unter der Annahme, dass hier jede Konfiguration im Konfigurationsraum, vergleiche Abschnitt 1.5.5, auch Pareto-effizient ist.

Durchgängiges Beispiel 3.2: Konfigurationsraum



Im Fokus des durchgängigen Beispiels ist die Tracking-Komponente mit impliziten Abhängigkeiten zu der Ressourcenplattform. Es ergeben sich vier gültige Systemkonfigurationen zur Umfelderfassung:

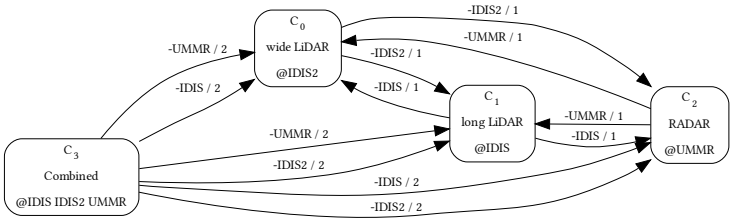
- C_0: Breite optische Erfassung mit LiDAR-Sensor *IDIS2*
- C_1: Weitreichende optische Erfassung mit LiDAR-Sensor *IDIS*
- C_2: Elektromagnetische Erfassung mit RADAR-Sensor *UMMR*
- C_3: Kombinierte Erfassung mit allen drei Sensoren

Eine konkrete Systemkonfiguration enthält dabei keine Variabilität und wird hier vereinfacht ohne die Kernkomponenten dargestellt. Weiterhin werden jeweils die qualitativen *Bewertungen* pro Analysedimension festgehalten.

3.3. Architekturrelationsbildung

Auf Basis der Artefakte des variantenreichen Entwurfs werden nachfolgend die Relationen zwischen den erhobenen Pareto-optimalen Konfigurationen analysiert und in einer Graphstruktur angeordnet, vergleiche Abschnitt 4.2. Der entstehende Architekturrelationsgraph gliedert den Entwurfsraum durch eine systematische Verknüpfung der Konfigurationen, vergleiche Abschnitt 4.1.1. In Abhängigkeit der eingesetzten Hardwareressourcen wird bestimmt wie Konfigurationen zueinander in Beziehung stehen. Weiterhin werden Rekonfigurationskosten erhoben und Aktionen definiert die Ressourcenausfälle abfangen, vergleiche Abschnitt 4.2.3. In Durchgängiges Beispiel 3.3 ist ein exemplarischer Architekturrelationsgraph skizziert.

Durchgängiges Beispiel 3.3: Architekturrelationsgraph



Der Architekturrelationsgraph zeigt die subtraktiven Zusammenhänge zwischen den Systemkonfigurationen auf. Zur kompakten Darstellung sind die Abhängigkeiten zur Ressourcenplattform als *@[Ressourcenname(n)]* notiert. An den Kanten der Form *-[Ressource]/[Kosten]* werden mögliche Ressourcenausfälle und die Rekonfigurationskosten annotiert. Die Kosten werden hier durch notwendige Umstellungen in der Ressourcenplattform erfasst. Hier wird davon ausgegangen, dass alle Ressourcen des Startknotens einer Kante vorhanden sind.

Ausgehende Kanten sind im initialen Architekturrelationsgraph inhärent nichtdeterministisch. Dies ist dadurch zu begründen, dass im Zuge der multikriteriellen Optimierung keine globalen Optima zur eindeutigen Abgrenzung von Konfigurationsalternativen identifiziert werden können. Es handelt sich um eine Vielfalt gültiger Lösungen im mehrdimensionalen Raum von Qualitäten. So stehen viele mögliche Alternativen zur Auswahl, die unterschiedliche Anforderungen an die Ressourcenplattform stellen und einzelne Qualitätsdimensionen wechselweise intensiv adressieren. Der Architekturrelationsgraph verweist auf diese Mengen von Konfigurationen. Zur weiteren Auswertung der qualitativen Erfüllung werden die entsprechenden Wertebelegungen der Dimensionen auf eine prozentuale Skala mit einem definierten Mindestwert normiert. In der Normierung dient jeweils das beste sowie das schlechteste Bewertungsergebnis pro Qualitätsdimensionen als Ober- bzw. Untergrenze. In Durchgängiges Beispiel 3.4 ergeben sich normierte Bewertungen zwischen vier ausgewählten Qualitätsdimensionen.

Durchgängiges Beispiel 3.4: Qualitätsdimensionen mit Werten

Konf.	Winkel	Reichweite	Robustheit	Verfügbarkeit
C_0	**1,00**	0,50	0,60	0,54
C_1	0,50	**1,00**	0,50	0,50
C_2	0,52	0,60	**1,00**	0,68
C_3	0,67	0,70	0,70	**1,00**

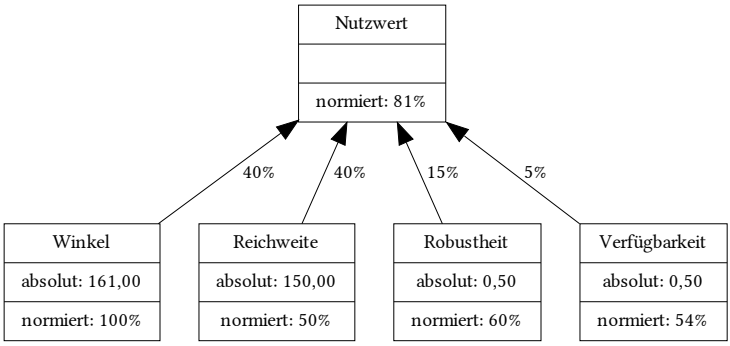
Auf Grundlage der Qualitätsdimensionen und Wertebelegungen aus Tabelle 1.1 in Abschnitt 1.5.6 wurde eine Normierung vorgenommen. Die Werte verlaufen zwischen einem definierten Minimum von 50% und der jeweiligen Bestbewertung von 100%.

Im allgemeinen Fall würde zur Festlegung einer Konfigurationsalternative ein hoher Aufwand zur Analyse aller ausgehenden Pfade entstehen. Durch Zielkonflikte zwischen den Analysedimensionen wäre dennoch nicht gewährleistet, dass eine Alternative eindeutig identifiziert wird. Erst konkrete Betriebsszenarien charakterisieren hinreichend welche Konfigurationen für die gewählte Aufgabe als angemessen beziehungsweise betriebsoptimal gelten. Im Prozess werden dahingehend die qualitativen Bewertungen in Form einer Evaluationsstruktur aufbereitet. Initial werden dabei alle Qualitäts-

dimensionen gleichmäßig balanciert gewichtet, vergleiche Abschnitt 4.2.4. Die entstandene Vorlage dient im Anschluss als visuelles Hilfsmittel zur *Kodierung von Betriebsszenarien* für jede Systemkonfiguration. Qualitätsdimensionen sind in realistischen Betriebsszenarien nicht gleichbedeutend. Der Anwender passt entsprechend pro Szenario methodisch die Gewichtungen an und definiert pro normalisierte Dimension minimale Akzeptanzwerte. Die Anpassungen gründen auf Geschäftsentscheidungen zur Identifikation einer optimalen Abwägung gegenläufiger Ziele. Zur Erläuterung zeigt Durchgängiges Beispiel 3.5 eine Evaluationsstruktur über die vormals definierten Qualitätsattribute und die Werte aus der Systemkonfiguration c_0 sowie einem exemplarischen Betriebsszenario.

Durchgängiges Beispiel 3.5: Evaluationsstruktur

Unter der Annahme, dass im Betriebsmodus *Annäherung* (vergleiche Abschnitt 1.5.1) insbesondere der Öffnungswinkel und die Reichweite wichtig sind und die Robustheit der Verfügbarkeit bevorzugt wird, werden folgende Gewichtungen durch den Anwender festgelegt: 40% Winkel, 40% Reichweite, 15% Robustheit und 5% Verfügbarkeit. Zur einfachen Darstellung werden keine minimalen Akzeptanzwerte für das Beispiel definiert. Die resultierende Instanz der Evaluationsstruktur führt zu einem akkumulierten Nutzwert von 81% für Systemkonfiguration C_0.

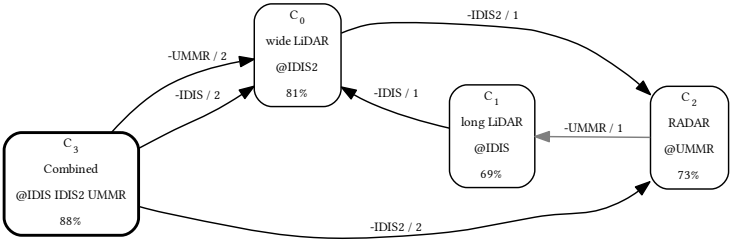


Die weiteren Nutzwerte sind 69% für C_1, 73% für C_2 und 88% für C_3.

Es resultiert pro Betriebsmodus eine Instanz der Evaluationsstruktur, die sich verkürzt als eine Produktsumme über normierte Wertebelegungen und Gewichtungen zur Nutzwertbestimmung beschreiben lässt. Die minimalen Akzeptanzwerte bilden dabei die Nebenbedingungen der Funktion. Für realistische Betriebsszenarien kann der Anwender weiterhin Zwischenschichten in der Struktur einfügen, um beispielsweise Analysedimensionen in direkte Beziehung zu setzen oder gruppenweise zu limitieren.

Durch die eindeutige Kodierung der Betriebsszenarien lässt sich ein deterministischer Architekturrelationsgraph pro Szenario erzeugen, vergleiche Abschnitt 5.1.2. Eine mögliche resultierende Reduktion eines deterministischen Architekturrelationsgraph unter Beachtung des Nutzwerts jeder Konfiguration wird in Durchgängiges Beispiel 3.6 aufgezeigt.

Durchgängiges Beispiel 3.6: Initialer Deterministischer Architekturrelationsgraph ohne Einfärbungen



Im resultierenden deterministischen Architekturrelationsgraph dürfen keine Mehrdeutigkeiten bestehen. Hierzu wird für jeden Ressourcenausfall nur eine Kante beibehalten, die in einen Knoten mit optimalen Nutzwerk führt. Dabei muss jedoch die Erreichbarkeit jedes Knotens weiterhin garantiert werden. So wird beim Ausfall von *UMMR* in C_2, nicht C_0, sondern der qualitativ nachteilige Knoten C_1 als Nachfolger festgelegt, vergleiche graue Markierung. Die Kante zwischen C_1 und C_0 bleibt nach dem Verfahren bestehen. An dieser Stelle identifiziert die Exploration keine gültige Alternative, da *IDIS2* auf jedem Pfad über C_1 bereits entfallen ist. C_3 dient als initiale Konfiguration für den aktuellen Betriebsmodus und besitzt den besten Nutzwert im Entwurfsraum sowie den höchsten Grad an Fehlertoleranz.

Mit Erstellung der DARG-Instanzen enden die vorbereitenden Analysen in der Entwurfszeit und das System geht in die Betriebsphase über. Dabei wird nach Festlegung des gewünschten Betriebsmodus eine *initiale Konfiguration* für den Beginn des Betriebs festgelegt. Diese Konfiguration bildet zugleich den Startknoten im betriebsspezifischen deterministischen Architekturrelationsgraph. Die gewählte Konfiguration nutzt entsprechend der Grapherzeugung die Ressourcen der Hardwareplattform teilweise bis vollständig, je nach Grad der Redundanzen im fehlertoleranten Systementwurf.

Im weiteren Verlauf wird von einer simulierten Betriebsphase ausgegangen. Dabei wird eine Konfiguration zwar identifiziert, jedoch weder in eine lauffähige Implementierung überführt, noch auf realer Hardware ausgeführt. Diese technische Verfeinerung liegt nicht im Fokus dieser Arbeit. Im realen Betrieb würde eine existierende Erkennung und Lokalisierung von Fehlern erfolgen, die eine Untermenge der Hardwareressourcen als defekt markiert und anschließend eine Rekonfiguration stimuliert. Nach der Identifikation einer alternativen Konfiguration würde eine Neuverteilung der Softwarekomponenten nach etablierten Verfahren durchgeführt werden. Weiterhin würde die Hardwareplattform auf elektronischer Ebene erneut kalibriert werden, um das System in einen operationalen Zustand zurückzuführen.

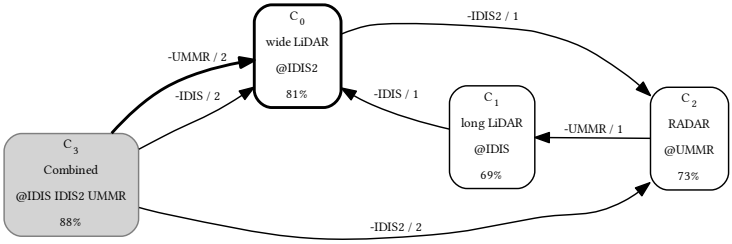
3.4. Alternativenexploration

In der Betriebsphase werden unter Annahme eines vorliegenden *Fehlermodells*, beispielsweise auf Basis einer Fehlerbaumanalyse, eine Fehlersequenz in das laufende System *injiziert*, vergleiche Abschnitt 5.2.1. Eine Fehlersequenz markiert mehrere Hardwareressourcen der Reihe nach als defekt. Falls eine entsprechende Ressource zur Ausführung der aktuellen Konfiguration eingesetzt wird, wird der zugehörige Knoten im Graphen als inaktiv markiert und die Alternativensuche gestartet.

Da auch eine Folgekonfiguration eines Knotens betroffen sein kann, werden vor der Auswahl einer identifizierten Alternative deren Ressourcenanforderungen überprüft. Trifft dies zu, wird mit dem jeweils nächsten Knoten in der Nachfolge fortgefahren. Sämtliche nicht-ausführbare Knoten werden eingefärbt und bei späteren Läufen übersprungen. Identifiziert die Suche nach Erreichen einer anwenderdefinierten Tiefe keine Alternative, wird

dies zurückgemeldet. Wird jedoch eine Alternative gefunden, wird diese zur Auswahl markiert und dem Anwender als *betriebsoptimale Alternativkonfiguration* vorgeschlagen, vergleiche Abschnitt 5.2.2. Unter der Annahme, dass die Ressource *UMMR* ausfällt, resultiert der eingefärbte deterministische Architekturrelationsgraph in Durchgängiges Beispiel 3.7. Hierbei wäre die Konfiguration C_0 die bevorzugte Alternative, erreichbar über den Pfad $C_3 \xrightarrow{-UMMR/2} C_0$.

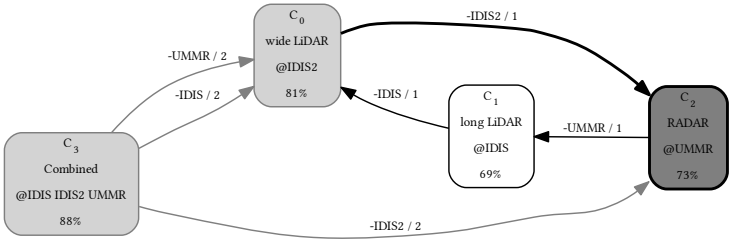
Durchgängiges Beispiel 3.7: Deterministischer Architekturrelationsgraph mit Einfärbungen (1)



Die ausfallende Ressource bewirkt das Auslösen einer Rekonfiguration hin zu C_0 und eine Hervorhebung des Zielknotens sowie der beteiligten Kante. Der gewählte Zustand erlaubt im weiteren Verlauf ein Wechseln direkt zu C_1.

Im aktuellen Zustand des durchgängigen Beispiels wäre nach dem Ausfall von *UMMR* nun die Ressource *IDIS2* im Betrieb und die Ressource *IDIS* im Stand-by. Entsprechend gibt es aus der Systemkonfiguration C_0 nur eine ausgehende Kante. Fällt *IDIS2* entsprechend aus, wird ein Wechsel nach C_2 erzwungen. Der resultierende Pfad $C_0 \xrightarrow{-IDIS2/1} C_2$ endet allerdings in einer bereits nicht mehr erfüllbaren Systemkonfiguration, vergleiche Durchgängiges Beispiel 3.8.

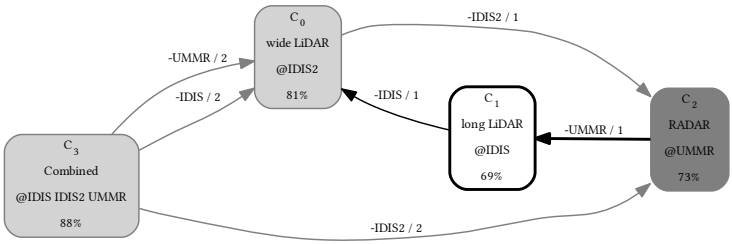
Durchgängiges Beispiel 3.8: Deterministischer Architekturrelationsgraph mit Einfärbungen (2)



Die ausgewählte Konfiguration erfordert *UMMR*, diese Ressource ist jedoch im Vorfeld ausgefallen. Entsprechend ist die Konfiguration C_2 nicht erfüllbar. Ein Wechsel nach C_1 ist ressourcenbedingt möglich und wird automatisiert durchgeführt.

Durch die automatisierte Fortsetzung der Suche konnte die valide Konfiguration C_1 unter Ausschluss von *UMMR* gefunden werden, vergleiche Durchgängiges Beispiel 3.9.

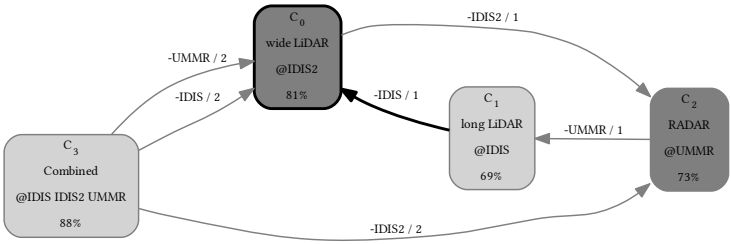
Durchgängiges Beispiel 3.9: Deterministischer Architekturrelationsgraph mit Einfärbungen (3)



Der vormalige Ausfall der Ressource *UMMR* bewirkt eine automatisierte Rekonfiguration hin zu C_1. Als verbleibende Folgekonfiguration ist C_0 erreichbar.

Der Ausfall der Ressource *IDIS* bewirkt das Verlassen der Konfiguration C_1 und bringt die Alternativenexploration zu einem abschließenden Endpunkt, vergleiche Durchgängiges Beispiel 3.10.

Durchgängiges Beispiel 3.10: Deterministischer Architekturrelationsgraph mit Einfärbungen (4)



Durch den Wegfall der Ressource *IDIS* wird die Konfiguration C_1 hin zu C_0 verlassen. Die Folgekonfiguration wurde im Vorfeld schon invalide und ist daher zu überspringen. Aus C_0 existiert jedoch keine ausgehende Kante. Die Exploration findet keine Alternative und das Verfahren endet.

Zur weiteren Dokumentation der Ereignisse werden dem Anwender weiterhin die Auswirkungen innerhalb der Entwurfsmodelle (geänderte Assemblierungen et cetera) sowie die resultierenden qualitativen und strukturellen Änderungen in Kennzahlen dargestellt.

4. Modellierung

Das Expertensystem zur Entscheidungsunterstützung in Auswahl einer adäquaten Konfigurationsalternative nutzt eine graphbasierte Struktur. In den folgenden Abschnitten werden die strukturellen Modelle und deren Erzeugung beschrieben.

4.1. Architekturrelationsgraph als ein Modell zur Entscheidungsunterstützung

Ein Architekturrelationsgraph trägt pro Knoten jeweils sämtliche relevanten Architektur- und Plattforminformationen zu einer Konfiguration. Konkret verweist ein Knoten neben der jeweiligen Ausprägung des Systemmodells weiterhin auf eine Teilmenge der Hardwareressourcen, die zur Ausführung der Systemvariante benötigt werden. Der Graph wird auf Grundlage einer unsortierten Konfigurationsmenge algorithmisch erzeugt. Die Konfigurationen stammen aus einer meta-heuristischen Vorselektion innerhalb des möglichen Entwurfsraums.

4.1.1. Strukturelle Spezifikationen

Strukturell betrachtet ist ein Architekturrelationsgraph als ein endlicher gewichteter *Digraph* mit schwach zusammenhängenden Knoten definiert. Die Spezifikation der Bestandteile der Architektur erfolgt durch das Palladio-Komponentenmodell (Englisch: Palladio Component Model, *PCM* [BKR09]). Das PCM beschreibt in der Basisvariante sechs partielle Metamodelle für jeweils spezialisierte Architektursichten, vergleiche [HJZ+17]. Die atomaren Softwarekomponenten und deren Schnittstellen werden durch das *Repository*

Model RM_{PCM} definiert. Die Softwarekomponenten werden dabei als Menge von Basiskomponenten (Englisch: *Basic Components*) beschrieben:

$$\mathcal{B} := \{c \in RM_{PCM}.getComponents() \mid c \in BasicComponents\}$$

Das Metamodell für die *Service Effect Specification* $SEFF_{PCM}$ spezifiziert ergänzend eine rudimentäre Verhaltensbeschreibung pro Komponente. In einem SEFF werden zugleich Anforderungen an ausführende Ressourcen quantifiziert. Die Integration ausgewählter Komponenten aus der Repository erfolgt in sogenannten Assemblierungskontexten (Englisch: *Assembly Contexts*) hin zu einer Softwarearchitektur und wird über das *System Model* SM_{PCM} beschrieben. Ausführende Ressourcen werden im Metamodell *Ressource Environment* RE_{PCM} entlang repräsentativer Eckdaten spezifiziert. Die Zuordnung von Komponenten auf Ressourcen erfolgt durch das *Allocation Model* AM_{PCM}. Das Benutzerverhalten wird im *Usage Model* UM_{PCM} unter Berücksichtigung der erwarteten Benutzerpopulation und deren Nutzungsintensität modelliert.

Die Variabilität innerhalb der PCM-Modellinstanzen wird durch Freiheitsgrade (Englisch: Degrees of Freedom, *DoF* [Koz11, KR11]) spezifiziert. Das Metamodell *Design Decisions* DD_{PCM} definiert dazu veränderbare Elemente innerhalb der Entwurfsmodelle, die ein Optimierungsalgorithmus entlang festgelegter Optionen explorieren und modifizieren kann. Ausgehend von einer Modellinstanz spannen die Freiheitsgrade einen Entwurfsraum mit Variationen der Ausgangsarchitektur, den sogenannten Lösungskandidaten, auf. Die produzierten Kandidaten werden auf Erfüllung (Englisch: *fitness*) gegenüber Anforderungen an spezifische Qualitätsdimensionen, beschrieben durch Qualitätsdeklarationen *qml declarations* QD_{PCM} (Quality Markup Language, QML[1]), gemessen. Die eindeutig definierten Qualitätsdimensionen werden nachfolgend als endliche Menge

$$Q := \{q_1 \dots q_n\} \mid q_i \in QD_{PCM}.getDimensions()$$

zusammengefasst. Genetische Modifikationsoperationen mutieren und überkreuzen iterativ die Lösungskandidaten, um einen hohen Grad an Variationen über viele Generationen von Populationen zu erreichen. Die Validität eines Lösungskandidatens wird durch $\mathbb{B}(F)$ als die Menge Boole'scher Formeln φ mit

$$\varphi := f \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \implies \varphi,\ f \in F$$

[1] Quality Markup Language (QML), Version 1.2.1, `http://www.qml-org.com`

beschrieben und überprüft. Aus den durch Genom-Kodierung insgesamt produzierten Kandidaten (Englisch: *all candidates*) ermittelt die Optimierung eine Menge Pareto-effizienter Kandidaten (Englisch: *achieved candidates*). Dabei werden in beiden produzierten Mengen weiterhin die Wertebelegungen jeder Analysedimension je Kandidat festgehalten.

Je nach vorliegenden Anwendungsfall werden systemspezifische Freiheitsgrade [ABG+13b] eingesetzt. Für den vorgestellten Ansatz werden spezifische Freiheitsgrade zur Selektion von alternativen Komponenten sowie zur variantenreichen Allokation von Komponenten auf ausführende Ressourcen genutzt.

Das *Ressource Environment* des Palladio-Komponentenmodells fokussiert sich auf die technische Beschreibung von CPU, Speicher und Festplatten von Serverknoten. Es ermöglicht jedoch keine Festlegung von Ein- und Ausschlussbeziehungen zwischen Ressourcen. Im entwickelten Ansatz stehen anstelle ausführenden Ressourcen vor allem die Sensorik und Aktuatorik eines Systems im Vordergrund. Die Ressourcen werden durch Kennzahlen aus Datenblättern funktional und qualitativ spezifiziert und bilden Redundanzgruppen. Diese Gruppen haben weiterhin strikte Abhängigkeiten zu anderen Gruppen und Einzelressourcen oder schließen diese exklusiv aus. Das PCM wird aus diesen Gründen um ein Metamodell zur Beschreibung der Ressourcenplattform nach Definition 4.1 für Sensorik und Aktuatorik ergänzt.

Definition 4.1: Ressourcenplattform

Sei Q eine endliche Menge von Qualitätsdimensionen und $\mathbb{B}(F)$ die Menge aller Boole'schen Formeln über φ. Eine Ressourcenplattform *RP* ist ein Tupel aus $\langle \mathcal{R}, spec, G, \phi \rangle$, wobei:

- $\mathcal{R} := \mathcal{R}_a \cup \mathcal{R}_s$ ist eine endliche Menge von Ressourcen mit:
  - $\mathcal{R}_a$ ist eine endliche Menge von Aktuatoren
  - $\mathcal{R}_s$ ist eine endliche Menge von Sensoren
  - $\mathcal{R}_a \cap \mathcal{R}_s = \emptyset$
- $spec : \mathcal{R} \times \mathbf{Q} \rightarrow \mathbb{R}_\perp$ liefert Spezifikationen für Ressourcen über der Menge aller Qualitätsdimensionen, wobei:

  - $\mathbb{R}_\perp := \mathbb{R} \cup \{\perp\}$
  - $\perp$ ist ein neutraler Wert, der die betrachtete Dimension unbelegt lässt
- $G \subseteq \mathcal{P}(\mathcal{R})$ sind Gruppen redundanter Ressourcen
- $\phi \subseteq \mathbb{B}(\mathcal{R} \cup G)$ sind Restriktionen auf Ressourcen und Redundanzgruppen

Die Ressourcenplattform beschreibt dabei neben den Ressourcenentitäten auch deren Eigenschaften und Beziehungen. Ressourcen werden als Sensoren oder Aktuatoren typisiert und mit technischen Spezifikationen belegt. Diese Spezifikationen bilden in den nachgelagerten Analysen die Daten über Auswahl und Wertbelegung der Qualitätsdimensionen ab. Durch die getrennte Definition der Spezifikationstypen als Dimensionen, wird eine mehrfache Wertebelegung des Typs pro Verwendung ermöglicht.

Gruppen dienen zur vereinfachten Festlegung von Redundanzen. Restriktionen zwischen Ressourcen werden innerhalb der Redundanzgruppen und optional zusätzlich gruppenübergreifend beschrieben. Für die Ein- und Ausschlussbeziehungen werden propositionale Ausdrücke entsprechend $\mathbb{B}(F)$ eingesetzt. Dabei kann die Auswahl expliziter Gruppen eine Auswahl weiterer Gruppen oder Einzelressourcen implizit fordern oder unterbinden. Die Implikationen können auf der rechten Regelseite beliebig tief in Konjunktionen, Disjunktionen und Negationen verschachtelt werden.

Nachfolgend werden unter Nutzung der eingeführten Modellelemente die grundlegenden Strukturen des Architekturrelationsgraphens und deren Zusammenhänge definiert. Grundlegend ist die Festlegung des Begriffs Systemkonfiguration nach Definition 4.2.

Definition 4.2: Systemkonfiguration

Sei $\mathcal{B}$ die Menge aller Basiskomponenten, $\boldsymbol{Q}$ die Menge aller Qualitätsdimensionen und $RP := \langle \mathcal{R}, spec, G, \phi \rangle$ eine Ressourcenplattform. Eine Systemkonfiguration SC ist ein Tupel aus $\langle id, B, bind, val, par \rangle$, wobei:

- id ist ein eindeutiger Identifikator einer Systemkonfiguration
- $B \subseteq \mathcal{B}$ ist eine endliche Menge assemblierter Basiskomponenten
- $bind\colon B \to \mathcal{P}(G)$ bindet Basiskomponenten mit Ressourcen R aus Redundanzgruppen G, wobei

$$\exists R \subseteq \mathcal{R}\colon\ (\forall r \in R : \exists b \in B\colon \exists g \in bind(b)\colon r \in g) \wedge R \models \phi$$

- $val\colon Q \to \mathbb{R}$ ist eine kumulierte Wertebelegung pro Qualitätsdimension
- $par \in \{\top, \bot\}$ beschreibt die Konfiguration als Pareto-effizient

Eine Systemkonfiguration umfasst die zum aktuellen Zeitpunkt assemblierten Basiskomponenten sowie deren zur Ausführung gebundenen Hardwareressourcen, unterteilt in Redundanzgruppen. Die Bindung von Ressourcen bezieht sich immer auf die Erfüllbarkeit von ϕ. Ist der betrachtete Lösungskandidat in der Menge Pareto-effizienter Kandidaten, wird die Konfiguration als Pareto-effizient markiert. Ausgehend von den produzierten Lösungskandidaten wird die Menge aller gültigen Systemkonfigurationen $\mathcal{SC}$ abgeleitet. Zur vereinfachten Schreibweise wird auf die referenzierten Basiskomponenten innerhalb einer Systemkonfiguration über die Funktion $bas\colon \mathcal{SC} \to \mathcal{P}(\mathcal{B})$ zugegriffen. Die endliche Menge aktuell gebundener Ressourcen aus den Redundanzgruppen einer Systemkonfiguration sei weiterhin über die Funktion $res\colon \mathcal{SC} \to \mathcal{P}(\mathcal{R})$ ableitbar.

Die Ermittlung der kumulierten Wertebelegungen pro Qualitätsdimension erfordert die Festlegung einer Funktion, die die Werte der einzelnen technischen Spezifikationen aus der Ressourcenplattform auswertet. Zur Erläuterung wird die folgende Interpretation herangezogen. Für eine Menge an Qualitätsdimensionen $Q' := \{q_1, q_2, q_3\}$ werden exemplarisch zwei Funktionen spezifiziert in der Form:

$$val(q_i \in Q') := \begin{cases} \sum_{r \in R} spec(r, q_i) & \text{für } i = 1 \\ \prod_{r \in R} spec(r, q_i) & \text{sonst} \end{cases}$$

Für eine Systemkonfiguration sc sei die gebundene Ressourcenmenge definiert als $res(sc) := \{r_1, r_2\}$ mit den exemplarischen Werten

$$spec(r_1, q_1) := 1 \quad spec(r_1, q_2) := 2 \quad spec(r_1, q_3) := \bot$$
$$spec(r_2, q_1) := 2 \quad spec(r_2, q_2) := 3 \quad spec(r_2, q_3) := 5$$

als technische Spezifikationen. In abgekürzter Schreibweise ergibt sich, unter Anwendung der Funktionen, für die betrachtete Systemkonfiguration

$$val(\forall q \in Q') = \begin{pmatrix} 3 \\ 6 \\ 5 \end{pmatrix}$$

als kumulierte Wertebelegung über alle Qualitätsdimensionen aus Q'.

Auf Grundlage der Systemkonfigurationen bildet der Architekturrelationsgraph (Englisch: Architecture Relation Graph, *ARG* [MN14]) die Wissensbasis für die Entscheidungsunterstützung ab. Dabei werden Konfigurationen entsprechend der Definition 4.3 in Relation zueinander gesetzt.

Definition 4.3: Architekturrelationsgraph

Sei $RP := \langle \mathcal{R}, spec, G, \phi \rangle$ eine Ressourcenplattform und $\mathcal{SC}(RP) = \{sc_1 \dots sc_n\}$ die Menge aller Systemkonfigurationen der Form $sc := \langle id, B, bind, val, par \rangle$ in Abhängigkeit zu *RP*. Ein Architekturrelationsgraph *ARG* ist ein Tupel $\langle SC, \rightarrow \rangle$ wobei:

- $SC \subset \mathcal{SC}$ ist eine endliche Menge von Systemkonfigurationen
- $\rightarrow \subseteq SC \times \mathcal{P}(T) \times \mathbb{N} \times SC$ ist eine gerichtete Distanzrelation zwischen einem Konfigurationspaar $sc_s, sc_t \in SC$ mit $\langle sc_s, T, c, sc_t \rangle \in \rightarrow$, wobei:
  1. $res(sc_s) \subseteq res(sc_t)$
  2. $T := res(sc_s) \setminus res(sc_t)$ beschreibt die Menge wegfallender Ressourcen nach einer Transition als trigger
  3. $c := |res(sc_s) \triangle res(sc_t)| + |bas(sc_s) \triangle bas(sc_t)|$ beschreibt die symmetrische Differenz gebundener

Ressourcen und referenzierter Basiskomponenten beider Konfigurationen als die Rekonfigurationskosten der Transition

Ein *ARG* repräsentiert den Entwurfsraum der produzierten Systemkonfigurationen und deren Zusammenhänge in einer Graphstruktur. Jeder Knoten repräsentiert eine Konfiguration mit einer inhärenten Abhängigkeit zu gebundenen Sensoren und Aktuatoren der Ressourcenplattform. Die Kanten des Graphen beschreiben die strukturellen Unterschiede zwischen Konfigurationen. Die daraus abgeleiteten Rekonfigurationskosten werden durch die symmetrischen Differenzmengen gebundener Ressourcen und eingesetzter Basiskomponenten der Architekturinstanz charakterisiert, siehe Abschnitt 4.2.2. Die Generierung des *ARG* ordnet und reduziert zugleich die Verbindungen innerhalb des Graphens, vergleiche Abschnitt 4.2.1. Dies verkleinert zeitgleich die Differenzmengen gebundener Ressourcen benachbarter Konfigurationen. Kanten des Graphen werden weiterhin mit einer Bedingung belegt, die im Falle einer annotierten wegfallenden Ressource ausgelöst wird. Im Folgenden wird eine Transition zwischen zwei Systemkonfigurationen sc_s und $sc_t \in SC$ innerhalb eines Architekturrelationsgraphens vereinfacht beschrieben als:

$$sc_s \xrightarrow{T/c} sc_t \ g.d.w. \ \langle sc_s, T, c, sc_t \rangle \in \rightarrow$$

Die Struktur ist initial nicht frei von Mehrdeutigkeiten, so sind die *trigger* ausgehender Kanten hochgradig nichtdeterministisch auszuwählen. Eine effiziente und transparente Entscheidungsunterstützung bedarf jedoch der deterministischen Auswahl.

Die Minimierung des *ARG* wird unter Verwendung der Resultate der Pareto-Optimierung durchgeführt. Ein unmittelbares Aufgreifen dieser eingeschränkten Konfigurationsmenge bereits bei der initialen Erstellung des *ARG* würde abgewertete, jedoch zugleich potenziell ressourcensparende, Konfigurationen verfrüht verwerfen. Eine qualitative Abwägung mehrerer Kanten, die den gleichen Ressourcenausfall behandeln, erfordert eine strikte Definition der Verhältnisse zwischen den Qualitätsdimensionen in $\mathbf{Q}$. Diese Gewichtung erfolgt im Zuge der Definition von Betriebsmodi. Weiterhin ist

eine Hierarchisierung der Dimensionen in Untergruppen durch den Anwender erstrebenswert. So wird neben der strukturellen Integrität der Systemkonfigurationen in $\mathcal{SC}$ die qualitative Erfüllbarkeit geltender Betriebsmodi verifiziert. Die Verifikation ist angelehnt an die Arbeiten von Florentz et al. [FH06] im Bereich Architekturevaluation, vergleiche Abschnitt 2.2.4. Eine Kodierung der Betriebsmodi wird in Form von gerichteten azyklischen Qualitätsattributsgraphen (Englisch: Quality Attribute Directed Acyclic Graph, *QADAG* [FH06]) nach Definition 4.4 durchgeführt.

Definition 4.4: Qualitätsattributsgraph

Sei $\mathcal{Q}$ eine endliche Menge von Qualitätsdimensionen und $SC := \langle id, B, bind, val, par \rangle$ eine Systemkonfiguration. Ein gerichteter azyklischer Qualitätsattributsgraph *QADAG* ist ein Tupel $\langle Q, \prec, norm, weight, wsum, eval \rangle$ wobei:

- $Q = Q_d \cup Q_s$ ist eine endliche Menge von Qualitätsattributen mit:
  - Q_d ist eine endliche Menge aus $\mathcal{Q}$ abgeleiteter Qualitätsattribute
  - Q_s ist eine endliche Menge strukturierender Qualitätsattribute
  - $Q_d \cap Q_s = \emptyset$
- $\prec \subseteq Q \times Q$ ist eine strenge Halbordnung über Q wobei:
  - $children \colon Q \to \mathcal{P}(Q), q \mapsto \{q' \in Q \mid q \prec q'\}$
  - $\exists! q_{root} \in Q \colon \{q \; inQ \mid q \prec q_{root}\} = \emptyset$ ist einfach gewurzelt
  - $\nexists\, q \in Q \colon q \prec children(q)$ ist azyklisch und irreflektiv
  - $\forall\, q \in Q_d \colon children(q) = \emptyset$ Elemente in Q_d sind Blattknoten
- *norm* normalisiert die Wertebelegung eines abgeleiteten Qualitätsattributs:

$$norm \colon Q_d \to \mathbb{R}, val(q) \mapsto [0; 1]$$

- $weight\colon Q \to \mathbb{R}, q \mapsto w$ ist die Gewichtung eines Qualitätsattributs
- *wsum* ist eine gewichtete Summe normalisierter Wertebelegungen:

$$wsum\colon \mathcal{P}(Q) \to \mathbb{R}, Q \mapsto \sum_{q \in Q} weight(q) * norm(q)$$

- *eval* wertet äußere und innere Knoten aus:

$$eval\colon Q \to \mathbb{R}, q \mapsto \begin{cases} wsum(q) & q \in Q_d \\ \sum_{q' \in children(q)} eval(q') & q \in Q_s \end{cases}$$

Ein *QADAG* ordnet eine endliche Menge von Qualitätsattributen in einer Graphstruktur an. Die Struktur ist analog zu einem einfach-gewurzelten Baum aufgebaut, erlaubt allerdings, dass äußere Knoten (Blattknoten) mehr als einen Vorgängerknoten besitzen. In äußeren Knoten werden alle untersuchten Qualitätsdimensionen aus **Q** als Qualitätsattribute festgehalten. Innere Knoten beschreiben strukturierende Qualitätsattribute und fassen die Werte ihrer Kindknoten zusammen. Bei der Instanziierung eines *QADAG* pro Systemkonfiguration werden diese Attribute mit den im *ARG* persistierten Messwerten belegt. Zur Kumulation der per se unvergleichlichen Attribute hin zu einer qualitativen Gesamtbewertung der betrachteten Konfiguration, werden die Belegungen auf Blattebene normalisiert. Die Normalisierung ermittelt alle Wertebelegungen aus den Analysen aller untersuchten Systemkonfigurationen und bildet diese auf einer Skala zwischen 0.00 und 1.00 ab. Sie lässt sich interpretieren als:

$$norm\colon Q_d \to \mathbb{R}, val(q) \mapsto \frac{v - v_{min}}{v_{max} - v_{min}}, \text{ wobei:}$$

- v die aktuell betrachtete Wertebelegung einer Dimension
- v_{min} ist das Minimum aller Wertebelegungen einer Dimension
- v_{max} ist das Maximum aller Wertebelegungen einer Dimension

In diesem Schritt wird zugleich die Einhaltung minimaler Akzeptanzwerte überprüft. Sollte eine definierte Unterschranke unterschritten werden, wird eine Normalisierung der betreffenden Belegung auf 0.00 durchgeführt.

Zusätzlich erhält jeder Knoten eine Gewichtung, die sich nach den Vorgaben des jeweiligen Betriebsmodus richtet. Die anschließende gewichtete Zusammenfassung der Werte erfolgt pro Unterbaum rekursiv aufwärts bis zum Wurzelknoten. In inneren Ebenen des Baums treten ausschließlich prozentuale Werte auf, die ohne weitere Konvertierung zueinander verrechnet werden können. Im Resultat bildet die Wurzel einer *QADAG*-Instanz die qualitative Gesamtbewertung der betrachteten Systemkonfiguration für einen spezifischen Betriebsmodus ab.

Nachdem die Kodierung der Betriebsmodi erfolgt ist, wird für jeden Modus ein deterministischer Architekturrelationsgraph (Englisch: Deterministic Architecture Relation Graph, *DARG* [MKR15]) entsprechend Definition 4.5 abgeleitet.

Definition 4.5: Deterministischer Architekturrelationsgraph

Sei $RP := \langle \mathcal{R}, spec, G, \phi \rangle$ eine Ressourcenplattform, $\mathcal{SC}(RP) := \{sc_1 \dots sc_n\}$ die Menge aller Systemkonfigurationen der Form $sc := \langle id, B, R, bind, val, par \rangle$ in Abhängigkeit zu RP und $ARG := \langle SC, \rightarrow \rangle$ ein Architekturrelationsgraph und $QADAG := \langle Q, \prec, nval, weight, wsum, eval \rangle$ ein Qualitätsattributsgraph. Ein deterministischer Architekturrelationsgraph *DARG* ist ein Tupel $\langle SC^*, \rightarrow^*, sc_a^*, mark, utility \rangle$ wobei:

- $SC^* \subset SC$ ist eine endliche Menge betriebsoptimaler Systemkonfigurationen
- $\rightarrow^* : \rightarrow \cap \langle SC^* \times \mathcal{P}(T) \times \mathbb{N} \times SC^* \rangle$ ist analog zu $\rightarrow$ eine gerichtete Distanzrelation zwischen einem betriebsoptimalen Konfigurationspaar
- $sc_a^* \in SC^*$ ist eine zum Betrieb ausgewählte betriebsoptimale Konfiguration
- $mark : SC^* \rightarrow \mathbb{R}, sc^* \mapsto [0; 1]$ liefert die Ausfallintensität der Ressourcen, die eine betriebsoptimale Konfiguration bindet

- $utility\colon SC^* \times Q \to \mathbb{R}, \forall val(q)\colon eval(q) \mapsto [0;1]$ liefert den Nutzwert einer betriebsoptimalen Konfiguration

Ein *DARG* bildet eine Verfeinerung eines *ARG* mit eindeutigen Kantenbedingungen und einer Markierung von Knoten. Weiterhin wird ein Knoten festgelegt, der die derzeit im Betrieb befindliche Konfiguration repräsentiert. Die Bedingungen der Kanten richten sich, analog zum *ARG*, nach den wegfallenden Ressourcen. Ein betriebsspezifischer *DARG* wird aus einem *ARG* in Verbindung mit einem *QADAG* instantiiert. Der *QADAG* wird mit den konkreten Belegungen der Qualitätsdimensionen parametrisiert um einen eindeutigen Nutzwert jeder mehrdeutig erreichbaren Konfiguration abzuleiten. Dieser Nutzwert wird zum direkten Vergleich der Konfigurationen eingesetzt um die jeweils nachteilig bewerteten Konfigurationen zu verwerfen, vergleiche Abschnitt 5.2.2. Sind situationsbedingt auch die Nutzwerte nicht unterscheidbar, werden die zu entfernenden Knoten randomisiert gewählt. Entsprechend der eingeführten Kurzschreibweise gelte hier analog für eine Transition zwischen zwei betriebsoptimalen Systemkonfigurationen sc_s^* und $sc_t^* \in SC^*$ innerhalb eines deterministischen Architekturrelationsgraphens:

$$sc_s^* \xrightarrow{T/c}{}^* sc_t^* \ g.d.w. \ \langle sc_s^*, T, c, sc_t^* \rangle \in \to^*$$

Ein *DARG* dient in der Simulation von Fehlerausfällen als dynamische Entscheidungsstruktur. Dies erfordert die nachträgliche Modifikation von Knotenmengen, die durch Ressourcenfehler betroffen sind. Eine unveränderte graphische Notation ist insbesondere relevant, um eine hohe Verständlichkeit und Übersicht in der Fehlerauswirkungsanalyse zu ermöglichen. Nur wenn das Layout des Graphen stabil bleibt, sind Änderungen des Entwurfsraums deutlich ersichtlich. Weiterhin würde das Streichen betreffender Knoten den Zusammenhalt des Graphen über Kanten zerstören. Konfigurationen, deren gebundene Ressourcen nach der Injektion eines Fehlers nicht mehr vollständig sind, werden im *DARG* daher farblich markiert, siehe Abschnitt 5.2.1. Die Einfärbung verläuft dabei entlang verschiedener roter Farbtöne. Dabei bildet die Helligkeit des Farbtons die Mächtigkeit der Menge der fehlenden Ressourcen ab. Ein hellroter Knoten ist nur minimal betroffen, ein dunkelroter entscheidend. Unabhängig von dem Grad der Einfärbung, werden markierte Knoten bei der Exploration immer übersprungen, siehe Abschnitt 5.2.3. Initi-

al ist ein deterministischer Architekturrelationsgraph vollständig frei von Einfärbungen.

4.1.2. Strukturelle Metamodellierung

Die definierten Kernelemente des Ansatzes werden in drei Metamodelle für die Ressourcenplattform, den Qualitätsattributsgraph und den (deterministischen) Architekturrelationsgraph überführt. Diese Metamodelle beschreiben die grundlegenden Strukturen für die Akquise und Aufbereitung der Daten.

Das Metamodell der Ressourcenplattform richtet sich nach der Definition 4.1. Aus Gründen der Übersichtlichkeit wird das Metamodell in zwei Schritten erläutert. Wie in Abbildung 4.1 notiert, erlaubt das Metamodell neben der Spezifikation von einzelnen Ressourcen und wiederverwendbarer technischer Eigenschaften weiterhin auch das Gruppieren der Ressourcen.

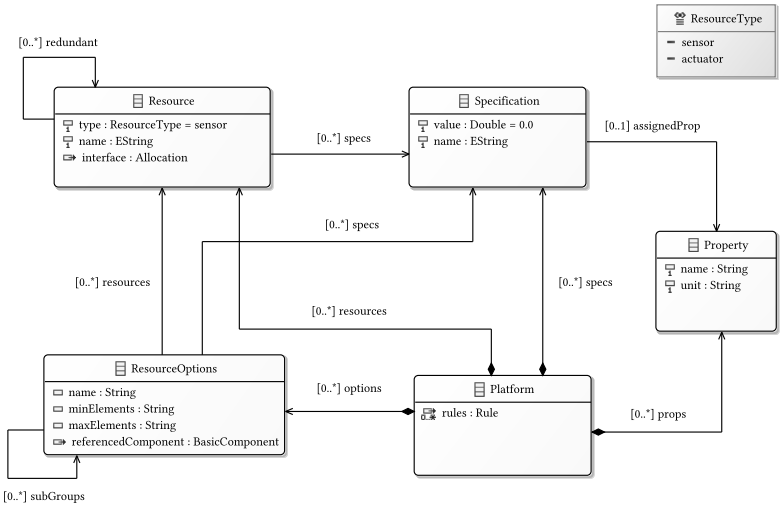


Abbildung 4.1.: Metamodell der Ressourcenplattform (Ressourcen)

In der Modellierung könnten so Gruppen genutzt werden, um redundante Ressourcen als Optionen zueinander anzuordnen. Durch die zusätzliche Angabe von minimalen und maximalen (benötigten) Elementen innerhalb der Gruppe, können kalte und heiße Redundanzen vereinfacht beschrieben werden. Zur Strukturierung von kontextabhängigen redundanten Ressourcen (zum Beispiel bei Abhängigkeiten durch physikalischen Einbauort im dreidimensionalen Raumkoordinatensystem) können Untergruppen genutzt werden.

Zusätzlich können sowohl über Gruppen als auch über einzelne Ressourcen Restriktionen in propositionalen Ausdrücken formuliert werden, siehe Abbildung 4.2.

Die Spezifikation erlaubt pro Regel das Formulieren von Implikationen einer Gruppe auf einen beliebig verschachtelten Ausdruck über weitere Gruppen beziehungsweise Ressourcen. In jedem Schritt sind Negationen zulässig.

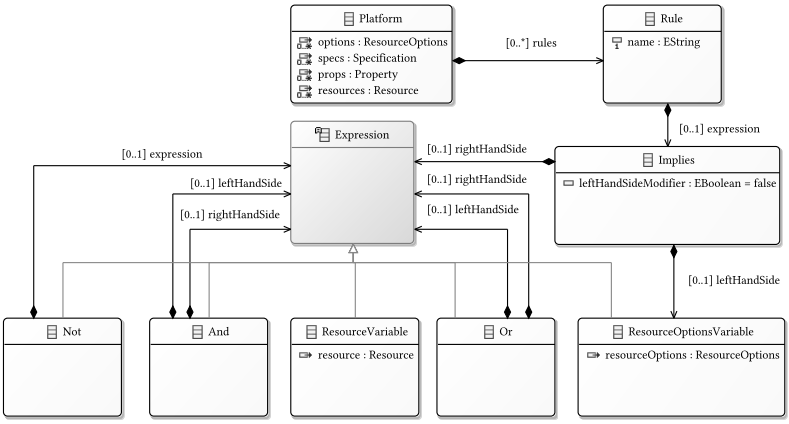


Abbildung 4.2.: Metamodell der Ressourcenplattform (Regeln)

Ausgehend von der Definition 4.4 beschreibt das Metamodell in Abbildung 4.3 die strukturellen Elemente eines Qualitätsattributsgraph.

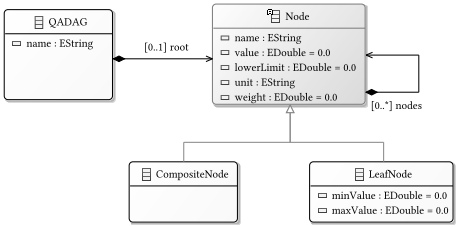


Abbildung 4.3.: Metamodell des Qualitätsattributsgraphens

Pro Instanz werden nach genau einem Wurzelknoten eine beliebige Anzahl hierarchisch geordneter Folgeknoten gefordert. Folgeknoten können als zusammengesetzte (innere) Knoten oder als Blattknoten auftreten. Alle Knoten enthalten als Daten Qualitätsattributsnamen, Wertebelegungen, minimale Akzeptanzwerte (unterste Grenzwerte), Werteeinheiten und Gewichtungen. In Blattknoten wird weiterhin das jeweilige Minima und Maxima aller gemessener Wertebelegungen der jeweils betrachteten Qualitätsdimension zur späteren Normalisierung konkreter Belegungen gesichert. Die zusammengesetzten Knoten sind rein strukturierend und akkumulieren ausschließlich die erhobenen Ergebnisse nachgelagerter Folgeknoten. Die definierten Funktionen auf dem Qualitätsattributsgraph werden in der Analysephase algorithmisch beschrieben.

Für beide Varianten des Architekturrelationsgraph wird ein gemeinsames Metamodell definiert, siehe Abbildung 4.4.

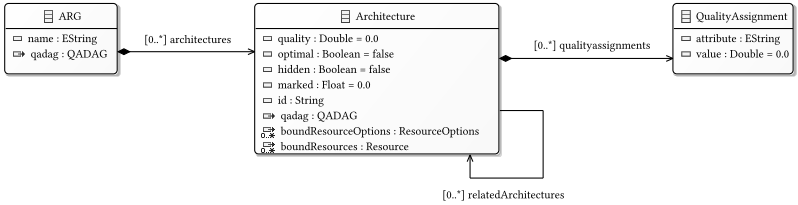


Abbildung 4.4.: Metamodell des Architekturrelationsgraphens

Abweichend von beiden Definitionen, werden Systemkonfigurationen direkt im Knoten gekapselt. Die Knoten des Graphen beschreiben so Architekturen, die auf eine Menge von Ressourcen in Redundanzgruppen verweisen. Im Zuge der Einbindung der Analysewerte aus der Pareto-Optimierung, kann jedem Knoten zusätzlich ein Satz an Qualitätsattributen mit zugehörigen Wertbelegungen zugewiesen werden. Die Kanten des Architekturrelationsgraphen werden im Metamodell durch gerichtete Relationen zu anderen Knoten beschrieben. Die zugehörigen „Trigger“ und Rekonfigurationskosten werden nach deren Ermittlung als attribuierte Kanten im Laufzeitmodell abgelegt. Der Graph manifestiert somit alle Daten, die in der Auswertung an einen angehängten betriebsspezifischen Qualitätsattributsgraphen übergeben werden. Zur Beschreibung des Zustands des deterministischen Architekturrelationsgraphens zur Laufzeit, ist es möglich die Knoten zu markieren. Diese Markierung ist, analog zur Definition 4.5, im kontinuierlichen Bereich möglich und erlaubt so eine Beschreibung der Markierungsintensität.

4.2. Algorithmische Spezifikation der Erzeugung der Graphen

Die Definition der formalen Strukturen und Metamodelle dienen ausschließlich als Grundlage für die Persitierung der Daten zur Entscheidungsunterstützung. Die Erzeugung valider Instanzen eines deterministischen Architekturrelationsgraphens bedarf der Aufbereitung der Daten aus dem Entwurfsraum. Daraus werden Knotenrelationen und Kantengewichte abgeleitet. Die nachfolgenden Prozessschritte werden zur Steigerung der Übersichtlichkeit in abgeschlossenen Algorithmen erläutert, die sich nach den vorhandenen Strukturen orientieren. Die technische Implementierung, vergleiche Abschnitt 6.6, fasst alle Schritte zur Reduktion der algorithmischen Komplexität zusammen.

4.2.1. Erzeugung und Parametrisierung der initialen Knotenmenge im Architekturrelationsgraphen

Zu Beginn des Prozesses wird je Architekturkandidat aus den Ergebnissen der vorgelagerten meta-heuristischen Entwurfsraumanalyse ein neuer

Knoten erzeugt. Dabei werden für die gewählten Optionen der Freiheitsgrade, die referenzierten Basiskomponenten aus der Instanz des Palladio-Komponentenmodells und deren gebundene Ressourcen aus der Ressourcenplattform ermittelt. Die erhobenen Informationen über die Systemkonfiguration werden im Knoten abgelegt. Jeder Knoten wird mit einem eindeutigen Bezeichner benannt und optional, entsprechend dem Ergebnis der Pareto-Optimierung, als optimal markiert. Zusätzlich werden alle Qualitätsdimensionen und zugehörige Wertebelegungen der jeweiligen Konfiguration im Knoten gespeichert. Anschließend wird jeder erzeugte Knoten in den *ARG* ohne ein- oder ausgehende Kanten eingetragen. Die Liste der Knoten ist zur weiteren Verarbeitung aufsteigend nach der Anzahl der gebundenen Ressourcen pro Systemkonfiguration sortiert.

Quelltext 4.1: Knotenerzeugung im Architekturrelationsgraphen

Daten : Menge der Architekturkandidaten *AC*, Qualitätsdimensionen **Q**, Wertebelegungen *V*, Ressourcenplattform *RP*
Resultat : *ARG*-Entwurf als lose sortierte Knotenmenge *arg*
Initialisierung der Knoten;
arg ← *new ARG()*;
foreach *ac* ∈ *AC* **do**
  node ← *new Node(ac.getId())*;
  node.initQualityValues(**Q**, *V*);
  foreach *dof* ∈ *ac.getAssemblyDofs()* **do**
    bc ← *dof.getAssembledBasicComp()*;
    node.addBasicComp(bc);
    foreach *resgrp* ∈ *RP.getResourceGroups()* **do**
      bc_{RP} ← *resgrp.getReferencedBasicComp()*;
      if *bc* = bc_{RP} **then**
        res ← *resgrp.getResources*();
        node.addResources(res);
        break;
      end
    end
  end
  node.setId(ac.getId());
  node.setOptimal(ac.isOptimal());
  arg.addNode(*node*);
end
Sortierung der Knoten nach Ressourcenmenge;
for *i* ← 1 . . . *arg.getNodes().size()* **do**
  j ← 1;
  $node_j$ ← *arg.getNode(j)*;
  $node_{j-1}$ ← *arg.getNode(j-1)*;
  while $j > 0 \wedge node_{j-1}.getResources().size() > node_j.getResources().size()$ **do**
    $node_{temp}$ ← $node_{j-1}$;
    arg.setNode(*j* − 1) ← $node_j$;
    arg.setNode(*j*) ← $node_{temp}$;
    j ← *j* − 1;
  end
end

Komplexitätsabschätzung: Für jeden Architekturkandidaten werden dessen Basiskomponenten mit den Komponenten in den Optionen aller Freiheitsgrade verglichen. Anschließend werden die erzeugten Knoten mittels des Bubblesort-Verfahrens nach der Ressourcenmenge absteigend sortiert. Für den Algorithmus resultiert ein Laufzeitverhalten von:

$$\begin{aligned} & O(n) * O(n) * O(n) + O(n^2) \\ =& O(n^3) + O(n^2) \\ =& O(n^3) \end{aligned}$$

4.2.2. Kostenorientierte Knotenrelationen im deterministischen Architekturrelationsgraphen

Die Relation der Knoten im *ARG* richtet sich nach den gebundenen Ressourcenmengen. Das Ausmaß der Veränderungen innerhalb der Auswahl von Basiskomponenten und Ressourcen bemisst somit die Ähnlichkeit der Konfigurationen. Die Ermittlung der Rekonfigurationskosten der Änderung entspricht der *cost*-Funktion nach Definition 4.3. Für jedes Paar aus Konfigurationen werden zunächst die symmetrischen Differenzmengen zwischen den gebundenen Ressourcen und Basiskomponenten ermittelt. Die Summe der Mächtigkeiten beider Mengen charakterisieren die Rekonfigurationskosten. Sofern die Kosten einen definierten *Schwellwert für Rekonfigurationskosten* nicht überschreiten, wird eine Kante zwischen dem Knotenpaar im Architekturrelationsgraph ergänzt und daran die Kosten als Kantengewicht abgetragen.

Die Ermittlung des Kantengewichts folgt, entsprechend der Situation zur Entwurfszeit, der konservativen Annahme, dass alle Ressourcen verfügbar sind und somit die größtmöglichen Differenzmengen existieren. Zur Laufzeit sinkt die Mächtigkeit dieser Mengen mit jeder ausfallenden Ressource rapide. Dadurch sinken die Rekonfigurationskosten ungleichmäßig im *ARG*. Je nach Zielsetzung der Kostenoptimierung und anfallender Kosten, ist eine dynamische Aktualisierung ergänzend zur Laufzeit vorzunehmen, vergleiche Abschnitt 5.2.1. Dabei ist jedoch eine Abwägung zwischen dem Aufwand der Aktualisierung und der Verbesserung der Voraussage durchzuführen.

Zugleich werden die Kanten um auslösende Aktionen (als Englisch: *trigger*) ergänzt. Diese Aktionen fangen den Ausfall von aktuell eingesetzten Ressourcen im Fehlerfall ab. Dazu werden genau die Hardwareressourcen annotiert, die der aktuelle Knoten repräsentiert, jedoch nicht dessen Folgeknoten.

Quelltext 4.2: Knotenrelationen im Architekturrelationsgraphen

Daten : *ARG*-Entwurf als lose sortierte Knotenmenge arg,
Rekonfigurationskostenschwellwert $cost_t$
Resultat : *ARG*-Entwurf als verknüpfte sortierte Knotenmenge arg
Knoten in Relationen setzen;
$nodes \leftarrow arg.getNodes()$;
foreach $node_s \in nodes$ **do**
  $bc_s \leftarrow node_s.getBasicComps()$;
  $res_s \leftarrow node_s.getResources()$;
  foreach $node_t \in nodes \setminus n_s$ **do**
    $bc_t \leftarrow node_t.getBasicComps()$;
    $res_t \leftarrow node_t.getResources()$;
    $irem \leftarrow bc_s \setminus bc_t$;
    $iadd \leftarrow bc_t \setminus bc_s$;
    $rrem \leftarrow res_s \setminus res_t$;
    $radd \leftarrow res_t \setminus res_s$;
    $costs \leftarrow |irem \cup iadd| + |rrem \cup radd|$;
    if $costs \leq cost_t \wedge rrem(node_s, node_t) \neq \emptyset$ **then**
      $e \leftarrow new\ Edge(node_s, node_t)$;
      $e.setCosts(costs)$;
      $e.setTrigger(rrem(node_s, node_t))$;
      $arg.addEdge(e)$;
    end
  end
end

Komplexitätsabschätzung: Es werden alle Architekturinformationen innerhalb der Knoten jeweils mit allen anderen Knoten verglichen. Dabei werden Mengenoperationen mit linearem und quadratischen Laufzeitverhalten nacheinander angewendet. Es resultiert ein Laufzeitverhalten von $O(n) * O(n) * O(n^2) = O(n^4)$.

4.2.3. Kostenorientierte Kantenreduktion im deterministischen Architekturrelationsgraphen

Die nachgelagerte Alternativensuche, vergleiche Abschnitt 5.2.1, erlaubt das temporäre Auswählen und Überspringen von Knoten im Architekturrelationsgraphen. Unter Berücksichtigung solcher *transienten Konfigurationen*, wird eine Reduktion redundanter Kanten durchgeführt. Dazu wird ausgehend vom Knoten mit der maximalen Menge gebundener Ressourcen eine Erreichbarkeitsanalyse aller Folgeknoten durchgeführt. Sobald ein echte Teilmengenbeziehung zwischen den gebundenen Ressourcen von mindestens drei Knoten gefunden wird, entfernt der Algorithmus die Kanten zwischen den zwei oder mehr äußeren Knoten aus dem *ARG*. Nebenbedingung für das Entfernen ist, dass die alternativen Pfade über die inneren transienten Konfigurationen keine höheren Rekonfigurationskosten verursachen.

Quelltext 4.3: Kantenreduktion im Architekturrelationsgraphen

Daten : *ARG*-Entwurf als verknüpfte sortierte Knotenmenge *arg*
Resultat : Kostenoptimaler *ARG arg*
Knoten reduzieren;
for $i \leftarrow 1 \ldots arg.getNodes().size() - 1$ **do**
  for $j \leftarrow i - 1 \ldots 1$ **do**
    $node_i \leftarrow arg.getNode(i)$;
    $node_j \leftarrow arg.getNode(j)$;
    $res_i \leftarrow node_i.getRessources()$;
    $res_j \leftarrow node_j.getRessources()$;
    if $res_i \subset res_j \wedge \exists k : i < k < j \mid res_i \subset arg.getNode(k).getRessources() \subset res_j$ **then**
      $costs_{i \rightarrow j} \leftarrow arg.getEdge(node_i, node_j).getCosts()$;
      $costs_{i \rightarrow k} \leftarrow arg.getEdge(node_i, node_k).getCosts()$;
      $costs_{k \rightarrow j} \leftarrow arg.getEdge(node_k, node_j).getCosts()$;
      if $costs_{i \rightarrow j} \geq costs_{i \rightarrow k} + costs_{k \rightarrow j}$ **then**
        $arg.removeEdge(node_i, node_j)$
      end
    end
  end
end

Komplexitätsabschätzung: Es werden alle Knoten (ohne einen Selbstvergleich) auf gemeinsame Ressourcenteilmengen untersucht. Dies führt zu einem quadratisch Laufzeitverhalten. Außerdem wird für jeden Vergleich eine transiente Architektur aus der Menge der Architekturen gesucht, die zwischen den zuvor ausgewählten Architekturen liegt. Diese Architektur wird auch auf Ressourcenteilmenge untersucht. Es ergibt sich ein Laufzeitverhalten von insgesamt $O(n) * O(n) * O(n^2) * O(n) * O(n^2) = O(n^7)$.

4.2.4. Erzeugung und Parametrisierung der strukturellen Vorlage des Qualitätsattributsgraphens

Die Ergebnisse der meta-heuristischen Pareto-Analyse enthalten Informationen zu den betrachteten Qualitätsdimensionen. Aus diesen Daten wird eine rudimentäre *QADAG*-Vorlage erzeugt. Die Vorlage bildet einen Qualitätsattributsgraphen mit flacher Hierarchie in zwei Ebenen. Nach Erzeugung eines Wurzelknotens wird für jede Qualitätsdimension ein neues Blatt erstellt und unmittelbar mit der Wurzel verbunden. Die Blätter sind initial gleichwertig gewichtet. Zur späteren Normalisierung der Belegungen werden aus den Analysedaten pro Dimension die jeweils minimalen und maximalen Wertebelegungen ermittelt und pro Blatt gespeichert. Minimale Akzeptanzwerte können aus den Daten nicht automatisiert erhoben

Quelltext 4.4: Initialisierung einer Qualitätsattributsgraphen-Vorlage

```
Daten : Qualitätsdimensionen Q, Wertebelegungen V
Resultat : QADAG-Vorlage qadag_t
Vorlage initialisieren;
qadag_t ← new QADAG();
root ← new Node("Nutzwert");
qadag_t.setRoot(root);
foreach quality ∈ Q do
    value_min ← ∞;
    value_max ← −∞;
    foreach value ∈ V.getValues(quality) do
        if value < value_min then
            value_min ← value;
        else if value > value_max then
            value_max ← value;
        end
    end
    node ← new Node(quality);
    node.setMinValue(value_min);
    node.setMaxValue(value_max);
    node.setWeight(1 / Q.size());
    root.addChild(node);
end
```

Komplexitätsabschätzung: Es wird für alle Qualitätsdimensionen der Wertebereich ermittelt, indem jeweils alle Wertebelegungen auf deren Extremwerte hin untersucht werden. Das Laufzeitverhalten ist dabei $O(n) * O(n) = O(n^2)$.

Nachdem die Strukturen und erzeugenden Algorithmen definiert sind, können für einen Architekturrelationsgraphen pro Betriebsmodus, jeweils kodiert als Qualitätsattributsgraph, *DARG*-Instanzen erzeugt und hinsichtlich Fehlerauswirkungen analysiert werden.

5. Analyse

Auf Grundlage der zuvor eingeführten Strukturen und erzeugenden Algorithmen für den kostenoptimalen *ARG* und den *QADAG*, wird nachfolgend die Instanziierung eines *deterministischen Architekturrelationsgraphens DARG* zur effizienten Analyse des Entwurfsraums beschrieben. Durch die Auswahl konkreter Betriebsmodi werden dabei adäquate betriebsoptimale Rekonfigurationsalternativen effektiv zur Laufzeit ermittelt. Die Analyse kann dabei zu unterschiedlichen Zeitpunkten im Entwicklungs- und Wartungsprozess erfolgen. Weiterhin ist eine ganzheitliche Berücksichtigung variierender Betriebsmodi möglich.

5.1. Grapheninstanziierung

Die Abgrenzung und Festlegung von *Trade-offs* ermöglicht die qualitative Abgrenzung von alternativen Systemkonfigurationen. Dabei ist es notwendig für jedes Betriebsszenario eine Instanz der im Abschnitt 4.2.4 erstellten *QADAG*-Vorlage zu bilden. Im Anschluss werden die vorhandenen Systemkonfiguration als Knoten im *ARG* ganzheitlich qualitativ bewertet und eine Menge betriebsspezifischer *DARG*-Instanzen abgeleitet. Zuletzt wird jeder resultierende deterministische Architekturrelationsgraph minimalisiert.

5.1.1. Nutzwertbestimmung durch Instanziierung und Auswertung eines Qualitätsattributsgraphens

Die Bestimmung von Nutzwerten lässt sich durch die Vergabe von Gewichtungen und minimale Akzeptanzwerte für die untersuchten Qualitätsattribute im Qualitätsattributsgraphen durch den Anwender umsetzen. Im vorgestellten Ansatz wird nach diesem Prinzip pro Szenario eine Instanz eines

Qualitätsattributsgraphens gebildet. Diese Instanz spiegelt die Anforderungen der Domäne mit Bezug zu den Qualitätsdimensionen der Pareto-Analyse wider.

Quelltext 5.1: Instanziierung eines Qualitätsattributsgraphens

Daten : *QADAG*-Vorlage $qadag_t$, Gewichtungen W, Akzeptanzwerte A
Resultat : *QADAG*-Instanz *qadag*
QADAG-Vorlage instanziieren;
$qadag \leftarrow qadag_t.clone()$;
foreach *node* ∈ *qadag.getRoot().getChildren()* **do**
    quality ← *node.getName*();
    node.setWeight(W.get(quality));
    node.setLowerBorder(A.get(quality));
end

Komplexitätsabschätzung: Nacheinander werden die Werte jedes Knotens in der *QADAG*-Vorlage initialisiert. Das Laufzeitverhalten ist dabei linear, also $O(n)$.

Die Instanzen mehrerer Qualitätsattributsgraphen können wahlweise durch den Anwender um weitere Hierarchieebenen erweitert werden. Dabei werden Qualitätsdimensionen in gewichteten Teilsummen zusammengefasst und gemeinsame minimale Akzeptanzwerte definiert. Im weiteren Verlauf dieser Arbeit wird von dieser Erweiterung abgesehen und eine flache Hierarchie aus einfacher Wurzel und einem Blattknoten je Analysedimension in den *QADAG*-Instanzen angenommen.

5.1.2. Instanziierung eines initialen deterministischen Architekturrelationsgraphens zur Analyse

Die Knoten im *ARG* beinhalten die Wertebelegungen der Analysedimensionen aus der Pareto-Optimierung und verweisen auf eine *QADAG*-Instanz. Die Werte werden im Zuge der Instanziierung des deterministischen Architekturrelationsgraphens in den referenzierten *QADAG* eingesetzt. Hierdurch lassen sich Nutzwerte für alle Systemkonfigurationen gemäß Definition 4.5

berechnen. Entsprechend resultiert für jede Instanz eines Qualitätsattributsgraphens auch eine *DARG*-Instanz. Innerhalb dieser Instanz wird abschließend durch den Anwender eine initiale Systemkonfiguration für den aktuellen Betrieb festgelegt. Die anfangs ausgewählte Systemkonfiguration bleibt über die gesamte Betriebsphase des Systems veränderbar.

Quelltext 5.2: Instanziierung eines deterministischen Architekturrelationsgraphens pro Qualitätsattributsgraph-Instanz

```
Daten : ARG arg, QADAG-Instanz qadag, ID der aktiven Betriebskonfigur.
        id_a
Resultat : DARG-Instanz darg
DARG instanziieren;
darg ← arg.clone();
foreach dnode ∈ darg.getNodes() do
    foreach qnode ∈ qadag.getRoot().getChildren() do
        quality ← qnode.getName();
        value ← dnode.getQualityValue(quality);
        if value ≠ ⊥ then
            qnode.setValue(value);
        end
    end
    QADAG instanziieren um Nutzwert zu ermitteln;
    result ← qadag.evaluate();
    dnode.setUtility(result);
    if dnode.getId() = id_a then
        darg.setActiveNode(dnode);
    end
end
```

Komplexitätsabschätzung: Auf Basis jeder Architektur im *ARG* werden die Werte für alle Knoten in der *QADAG*-Instanz gesetzt. Das Laufzeitverhalten entspricht $O(n) * O(n) = O(n^2)$.

Auch im aktuellen Stadium ist der *DARG* potenziell mit Mehrdeutigkeiten behaftet. Zur eindeutigen Auswahl einer Konfigurationsalternative ist eine Minimierung des Graphens unter Berücksichtigung der Nutzwerte notwendig. Da ein Knoten jedoch im Allgemeinen mehrere Vorgängerknoten haben

kann, ist eine Minimierung erst dann eindeutig bestimmbar, nachdem ein Fehler auftritt. Ein Ressourcenfehler stimuliert die Auswahl eines Teilpfades im *DARG* und kann somit auch Mehrdeutigkeiten provozieren. Um eine Parametrisierung über alle möglichen Fehler – maximal die Gesamtmenge aller Ressourcen innerhalb der Ressourcenplattform– zu vermeiden, wird diese Minimierung als Teil der Alternativenexploration im Graphen durchgeführt.

5.2. Alternativenexploration zur Laufzeit

Nach den vorbereiteten Berechnungen und der Instanziierung des Qualitätsattributsgraphens sowie des deterministischen Architekturrelationsgraphens, geht die Analyse in die Laufzeitphase über. In dieser Phase werden die Auswirkungen von Ressourcenfehlern beurteilt und alternative Konfigurationen gesucht. Als Resultat der Vorarbeiten stehen minimierte *DARG*-Instanzen für eine effiziente Fehleranalyse für vordefinierte Betriebsszenarien bereit. In Folge des Eintretens eines Fehlerfalls werden innerhalb eines deterministischen Architekturrelationsgraphens alle geeigneten Alternativen erhoben und qualitativ priorisiert. Die Fehler wirken sich auf die Größe des effektiven Entwurfsraums aus. Veränderungen des Raums werden im Verbund mit den verbleibenden möglichen Pfaden zur Rekonfiguration analysiert und zur Entscheidungsunterstützung für den Anwender aufgearbeitet. Die nachfolgenden Analyseschritte werden pro Ressourcenfehler nacheinander iterativ durchgeführt, um eine Beobachtung unter Auswirkungen der tatsächlichen Auftrittsreihenfolge der einzelnen Fehler zu beobachten.

5.2.1. Fehler- und Auswirkungsanalyse

Das Auftreten eines Fehlers impliziert den Ausfall mindestens einer Ressource aus der Ressourcenplattform. Enthält die Menge der in der aktuellen Konfiguration gebundenen Ressourcen diese fehlerhafte Ressource, wird der jeweilige Knoten im *ARG* markiert. Es resultiert ein fehlermarkierter deterministischer Architekturrelationsgraph, der zur weiteren Analyse genutzt wird.

Quelltext 5.3: Relevanzbeurteilung für Fehlerinjektion

```
Daten : DARG-Instanz darg, Fehlerhafte Ressource r_d
Resultat : Fehlermarkierte DARG-Instanz darg
Relevanz des Fehlers beurteilen und darg markieren;
foreach node ∈ darg.getNodes() do
    foreach res ∈ node.getResources() do
        if res = r_d then
            marking ← node.getMarking();
            node.setMarking(marking + 1);
        end
    end
end
```

Komplexitätsabschätzung: Für jede Architektur im *DARG* werden alle gebundenen Ressourcen mit den ausgefallenen Ressourcen innerhalb der Ressourcenplattform verglichen. Daraus resultiert ein Laufzeitverhalten von $O(n) * O(n) * O(n) = O(n^3)$.

5.2.2. Qualitätsorientierte Knotenreduktion

Wie Abschnitt 4.1.1 beschreibt, wurde von einem frühzeitigen Ausschluss Pareto-ineffizienter Kandidaten abgesehen, um von einem reichhaltigen Entwurfsraum zur Rekonfiguration zu profitieren. Somit sind sämtliche validen Lösungskandidaten der Pareto-Analyse in der aktuellen Instanz des *DARG* enthalten. Dies ermöglicht eine Alternativenauswahl auch dann noch, wenn eine favorisierte Pareto-effiziente Konfiguration fehlerhaft ist.

Trotz der kostenorientierten Reduktion in Abschnitt 4.2.3 kann bisher nicht garantiert werden, dass sämtliche ausgehende Kanten im *DARG* frei von Nichtdeterminismus sind. So können Geschwisterknoten mit identischem Vorgänger potenziell die gleichen Kantengewichte und *trigger* besitzen. Dies würde in der Graphexploration die automatisierte Auswahl einer Alternativkonfiguration verhindern. Durch die Festlegung der Betriebsanforderungen können jedoch im nächsten Analyseschritt mehrdeutige Segmente auf die Dominanz von ganzheitlichen Nutzwerten untersucht und abgrenzt werden.

Weiterhin lassen sich Konfigurationen, die mindestens einen minimalen Akzeptanzwert verletzen und somit einen Nutzwert von 0, 00 besitzen, herausfiltern. So ist neben der Optimierung von Rekonfigurationskosten eine Qualitätsoptimierung umsetzbar. Suboptimale Knoten können im Graphen so gekennzeichnet werden, dass diese in der Exploration eine nachgelagerte Rolle spielen. Aus Gründen der Effizienz wird die qualitative Reduktion nur für einen von Fehlern betroffenen Subgraph, das heißt für alle direkt und indirekt nachfolgenden Knoten, durchgeführt. Sobald mindestens ein valider Nachfolger identifizierbar ist, wird die Suche vorzeitig beendet.

Ein Knoten wird in einer *DARG*-Instanz *verborgen* (Englisch: *hidden*, temporär ausgeblendet), wenn er von mindestens einem Geschwisterknoten dominiert wird. Eine notwendige Nebenbedingung ist, dass der bevorzugte Knoten nicht fehlermarkiert ist. Wird diese Bedingung verletzt, wird der jeweilige Knoten als mögliche Lösung ignoriert und übersprungen. Nach diesem Prinzip werden alle weiteren Geschwisterknoten schrittweise in absteigender Reihenfolge ihrem Nutzwert nach verarbeitet. Die Suchtiefe ist aus den vorherigen Festlegungen der maximalen Rekonfigurationskosten, vergleiche Abschnitt 4.2.3, infolgedessen auch begrenzt.

In Abhängigkeit des fehlermarkierten *DARG* und der aktuell zum Betrieb ausgewählten Systemkonfiguration, initial $darg.getActiveNode()$, wird zunächst überprüft, ob die aktuelle Konfiguration unmittelbar von dem Ressourcenausfall betroffen ist. Falls dies zutrifft, werden alle Folgeknoten des ausgefallenen Knotens nach den genannten Kriterien rekursiv inspiziert. Wird in der ersten Ausführung kein gültiger direkter Folgeknoten zur Rekonfiguration identifiziert, untersucht die darauffolgende Ausführung den *DARG* auf gültige indirekte Folgeknoten in der nächsten Suchtiefe. Dabei selektiert der Algorithmus aus der Liste der zuvor identifizierten Nachfolger den Knoten mit dem höchsten Nutzwert als neuen aktiven Knoten aus und startet die nächste Rekursion. Dieses Vorgehen wird bis zum Erreichen der maximalen Graphentiefe ab dem jeweils aktiven Knoten fortgesetzt.

Aus Darstellungsgründen wird der Algorithmus in zwei Segmenten beschrieben. Im ersten Teil wird die Bestimmung der Nachfolgeknoten umgesetzt, im zweiten Teil das Verbergen dominierter Knoten.

Quelltext 5.4: Knotenreduktion im deterministischen Architekturrelationsgraphen nach Nutzwert der Konfigurationsalternativen

```
Daten : Fehlermarkierte DARG-Instanz darg, initale aktive Konfiguration
        node_a
Resultat : Subgraph der DARG-Instanz darg
Prozedur für rekursive Analyse;
Procedure reduceSuccessors(node):
    followingEdges ← new List<Edge>();
    Relevanz beurteilen;
    if node.getMarking() > 0 then
        Kanten zu Nachfolgeknoten identifizieren;
        utility ← 0;
        node_u;
        foreach edge ∈ darg.getEdges() do
            node_s ← edge.getSource();
            if node_s = node then
                node_t ← edge.getTarget();
                if node_t.getMarking() = 0 then
                    followingEdges.add(edge);
                    Knoten mit höchsten Nutzwert speichern;
                    if node_t.getUtility() ≥ utility then
                        utility ← node_t.getUtility();
                        node_u ← node_t;
                    end
                end
            end
        end
        if followingEdges.size() = 0 then
            followingEdges ← reduceSuccessors(node_u);
        end
    end
return followingEdges;

Kanten im DARG reduzieren und neu setzens;
Erste Rekursion mit initialen Knoten starten;
darg.setEdges(reduceSuccessors(node_a));
```

Komplexitätsabschätzung: Der Algorithmus iteriert über alle Kanten einer *DARG*-Instanz und führt eine Rekursion auf den nicht-markierten Nachfolgeknoten mit dem besten Nutzwerten aus. Das Laufzeitverhalten innerhalb einer Rekursion ist $O(n)$. Da der Graph azyklisch ist, kann maximal n-mal eine Rekursion auftreten. Im schlimmsten Fall resultiert somit ein Laufzeitverhalten von $O(n^2)$.

Quelltext 5.5: Knotenverbergen im deterministischen Architekturrelationsgraphen nach Rekonfigurationskosten

```
Daten : Subgraph der DARG-Instanz darg
Resultat : Reduzierte DARG-Instanz darg
followingEdges ← darg.getEdges();
Dominierte Knoten verbergen;
foreach fedge1 ∈ followingEdges do
    trigger1 ← fedge1.getTrigger();
    costs1 ← fedge1.getCosts();
    fnode1_t ← fedge1.getTarget();
    foreach fedge2 ∈ followingEdges \ fedge1 do
        trigger2 ← fedge2.getTrigger();
        costs2 ← fedge2.getCosts();
        fnode2_t ← fedge2.getTarget();
        if ¬fnode2_t.isHidden() ∧ trigger1 = trigger2 ∧ costs1 = costs2 then
            if fnode1_t.getUtility() > fnode2_t.getUtility() then
                fnode2_t.setHidden();
            else
                fnode1_t.setHidden();
            end
        end
    end
end
```

Komplexitätsabschätzung: Alle Kanten in der *DARG*-Instanz werden mit den jeweils anderen Kanten im Graphen auf Gleichheit der Rekonfigurationskosten und Trigger verglichen. Es folgt ein Laufzeitverhalten über $O(n) * O(n) = O(n^2)$.

5.2.3. Visuelle Entscheidungsunterstützung

Die Entscheidungsunterstützung für den Anwender basiert auf dem reduzierten *DARG*. Wirkt sich ein Ressourcenfehler negativ auf die aktuelle Konfiguration aus, wird der Graph entsprechend der zuvor erhobenen Daten aufbereitet.

Quelltext 5.6: Knotenkolorierung im deterministischen Architekturrelationsgraphen nach Pareto-Effizienz und Verfügbarkeit

```
Daten : Reduzierte DARG-Instanz darg, aktive Konfiguration node_a
Resultat : Eingefärbte DARG-Instanz darg
Relevanz beurteilen;
if node_a.getMarking() > 0 then
    col ← new Color(GREY);
    node_a.setColor(col);
    Farbton zur Kolorierung ermitteln;
    foreach node ∈ darg.getNodes() \ node_a do
        marking ← node.getMarking();
        if marking = 0 then
            if node.isOptimal() then
                if node.isHidden() then
                    col ← new Color(DARKGREEN);
                else
                    col ← new Color(GREEN);
                end
            else
                if node.isHidden() then
                    col ← new Color(DARKBLUE);
                else
                    col ← new Color(BLUE);
                end
            end
        end
        Knoten kolorieren (Standardfarbton ist GREY);
        node.setColor(col);
    end
end
```

Betroffene Knoten werden eingefärbt und mögliche Rekonfigurationspfade zu einer *betriebsoptimalen Alternativkonfiguration* hervorgehoben. Die visuelle Datenaufbereitung liefert zwei Darstellungen des *DARG*. Die Kolorierung sieht in den beiden Auswertungsstufen ein mehrstufiges Farbspektrum vor.

In der Detailansicht des Graphens wird der gesamte Raum an Entwurfsalternativen aufgezeigt, der sich nach Auftreten eines Fehlers ergibt. Fehlermarkierte Konfigurationen werden *grau* dargestellt. Gültige Konfigurationen werden in *grün* oder *blau* eingefärbt, ersteres trifft auf eine Pareto-effiziente Alternative zu, letzteres auf eine Pareto-ineffiziente. Verborgene (dominierte) valide Konfigurationen werden analog in einem dunklen Grün- beziehungsweise Blauton dargestellt, um eine Zweitrangigkeit bei der Alternativenfindung zu markieren. Eine grüne Konfiguration deutet auf einen maximalen Nutzwert für das gewählte Betriebsszenario hin.

Komplexitätsabschätzung: Die aktive Konfiguration wird untersucht und es werden die Farbwerte für alle Knoten in der *DARG*-Instanz erneut ermittelt. Das Laufzeitverhalten liegt demnach bei $O(n)$.

In der zweiten Stufe der Datenaufbereitung wird der Pfad zur betriebsoptimalen Alternativkonfiguration gesondert hervorgehoben. Alle weiteren Elemente um den Pfad herum werden ausgeblendet. Die Exploration sucht in Folge eines Fehlers mit relevanten Auswirkungen auf die aktuelle Konfiguration eine gültige Alternativkonfiguration. Diese wird durch die Vorverarbeitung im Graphen auf einem kostenoptimalen Weg erreicht. Die Suche lässt es explizit zu, dass invalide oder ineffiziente Architekturen übersprungen werden. Dafür wird bei der Auswahl einer Konfiguration immer auch ein Weitersprung in eine nachgelagerte Konfiguration auf einen qualitativen Mehrwert überprüft. Im Idealfall findet die Exploration eine gültige Systemkonfiguration, die entweder Pareto-effizient ist oder zumindest noch valide. Endet der Pfad hingegen ohne das vorherige Erreichen einer grünen oder blauen Konfiguration in einer fehlermarkierten Alternative, wird dieser Knoten rot eingefärbt. In diesem Fall ermittelt die Exploration des *DARG* keine gültige Alternative. Ein Pfad verläuft über eine Sequenz der auftretenden Fehler über die Knoten des deterministischen Architekturrelationsgraphens.

Quelltext 5.7: Erhebung und Kolorierung eines Rekonfigurationspfads im deterministischen Architekturrelationsgraphen

Daten : Eingefärbte *DARG*-Instanz *darg*, aktive Konfiguration $node_a$
Resultat : Eingefärbte *DARG*-Instanz *darg* mit Pfadhervorhebung
Procedure `checkAlternatives(`*node*, *searchHidden*`)`**:**
    Den aktuellen Knoten hervorheben und Alternative suchen;
    node.setHighlight();
    foundAlternative $\leftarrow \bot$;
    foreach *edge* $\in$ *darg.getEdges()* **do**
        $node_s \leftarrow$ *edge.getSource()*;
        if $node_s$ = *node* **then**
            $node_t \leftarrow$ *edge.getTarget()*;
            if $node_t$*.getMarking()* $= 0 \wedge$ $node_t$*.isHidden()* = *hidden* **then**
                foundAlternative $\leftarrow \top$;
                $node_t$*.setHighlight()*;
            else
                foundAlternative $\leftarrow$ `checkAlternatives(`$node_t$, *hidden*`)`;
            end
            if $\neg$*foundAlternative* **then**
                Keine lokale valide Folgekonfiguration; Knoten rot einfärben;
                $node_t$*.setColor(new Color(RED))*;
            end
        end
    end
return ***foundAlternative***;

Suche nach Pareto-effizienten Konfigurationen beginnen;
if $\neg$`checkAlternatives(`$node_a$, $\bot$`)` **then**
    Suche fortsetzen nach Pareto-ineffizienten Konfigurationen;
    if `checkAlternatives(`$node_a$, $\top$`)` **then**
        Keine globale valide Folgekonfiguration; Knoten rot einfärben;
        $node_a$*.setColor(new Color(RED))*;
    end
end

Komplexitätsabschätzung: Die Prozedur iteriert über alle Kanten in der eingefärbten *DARG*-Instanz und führt eine Rekursion mit einem Zielknoten durch, wenn die Kante ungültig ist. Die Anzahl der Aufrufe entspricht der Knotenanzahl im *DARG*. Der Algorithmus ruft die rekursive Prozedur maximal zweimal auf. Daraus resultiert ein maximales Laufzeitverhalten von $2 * O(n) * O(n) = O(n^2)$

Neben der Visualisierung werden eine Reihe von statistischen Kennzahlen zur Unterstützung für den Anwender erhoben.

5.2.4. Kennzahlen zur Unterstützung und Dokumentation von Rekonfigurationsentscheidungen

In der Analyse werden diverse Kennzahlen erhoben. Die Zahlen geben einerseits Auskunft zu dem Fehlerzustand des Systems und andererseits über dessen verbleibende Konfigurierbarkeit. Die Interpretation der Kennzahlen wird in den nachfolgenden Abschnitten vertieft. Die folgenden Kennzahlen charakterisieren den Systemzustand nach jedem auftretenden Fehler.

5.2.4.1. Aggregierte Kennzahlen

Eine Liste über *alle Ressourcen der Ressourcenplattform* beschreibt den Idealzustand der Systemhardware. Darauf bezogen wird die aktuell *fehlermarkierte Ressource* festgehalten. Unter Berücksichtigung aller vorherigen Ressourcenfehler, wird eine Liste über *alle noch verfügbaren Ressourcen* ermittelt. Die *Anzahl der gültigen Knoten* (Konfigurationen) im *DARG* beschreibt die Entwurfsraumgröße. Der Grad der Vermaschung wird als *Anzahl der Kanten* im *DARG* festgehalten.

5.2.4.2. Graphbasierte Kennzahlen

Aus dem *DARG* werden zweidimensionale Kennfelder über Knotenpaare erhoben. Spalten entsprechen den Quellkonfigurationen, Zeilen den Zielkonfigurationen. Einige Zellen im Kennfeld können unbelegt bleiben, da eine vollständige Vermaschung im *DARG* durch die Minimierung vermieden wird. Einerseits werden die *Rekonfigurationskosten* und die *Änderungen der*

Gesamtqualität zwischen Knotenpaaren festgehalten. Die Kennzahlen geben Aufschluss über die strukturelle und qualitative Entwurfsraumdichte. Im Eindimensionalen wird für jede Konfiguration festgehalten wie viele *Folgekonfigurationen* maximal über ausgehende Kanten noch erreichbar sind. Diese Kennzahl beschreibt die möglichen Maximallängen von Rekonfigurationspfaden ohne qualitative Präferenzen. Sie konkretisiert die Adäquatheit des Modells in der Repräsentation eines realistischen Entwurfsraums.

5.2.4.3. Pfadbasierte Kennzahlen

Folgt aus dem aktuell betrachteten Ressourcenausfall eine gültige Rekonfiguration, werden folgende Kennzahlen im Bezug auf den Rekonfigurationspfad festgehalten. Ein Pfad entspricht dabei einer Aneinanderreihung von Segmenten im deterministischen Architekturrelationsgraphen. Eindimensional werden die *Anzahl der Knoten im Pfad* sowie die *Summe aller ausgehenden Kanten* die im Suchprozess verglichen wurden, das heißt die Anzahl notwendiger Qualitätsvergleiche, festgehalten. Beide Kennzahlen dienen als ein Indikator zur Beurteilung der Effizienz der Suche. Als Kennfelder zwischen Quell- und Zielkonfiguration werden die *Änderungsaufwände* und *qualitative Änderungen* in *Konfigurationskosten auf dem Pfad* von der aktuellen Konfiguration zur alternativen Lösung schrittweise festgehalten. Diese Wertepaare nutzen der Beurteilung der Effektivität der Exploration. Ebenfalls als Kennfeld über die Konfigurationspaare im Pfad, wird die qualitative Streuung bei allen ausgehenden Kanten ermittelt, das heißt die *qualitativen Differenzen die als Optionen* bei der Auswahl einer ausgehenden Kante vorliegen. Die Werte helfen bei der Quantisierung der Degradation im Verlauf der Rekonfiguration.

5.3. Verwendungen der Analyseergebnisse in Entwurf und Betrieb

Die instanziieren Modelle sind im gesamten Lebenszyklus eines Systems einsetzbar. So lassen sich die Ergebnisse der Analyse in der Wartung in einem synchronisierten Offline-Wartungssystem einspeisen und dort zur

Beobachtung und Planung von manuellen Aktivitäten am Produktivsystem nutzen. Alternativ ist eine nachgelagerte Verarbeitung im produktiven System umsetzbar. Dies setzt jedoch eine vollständige Implementierung aller Systemkonfigurationen im *DARG* voraus. Abschließend können die Ergebnisse durch die leichtgewichtige Modellierung ebenfalls in frühen Entwurfsphasen des Systems zur Optimierung des Konfigurationsraums genutzt werden.

Je nach Zeitpunkt der Anwendung wird Personal aus Entwicklung beziehungsweise Wartung für die Entscheidungsfindung hinzugezogen. Eine wesentliche Herausforderung ist die Antizipierbarkeit von Fehlern und das Vorhalten von ausführbaren alternativen Systemkonfigurationen.

5.3.1. Selbstmanagement im Lebenszyklus des Systems

Der vorgestellte Ansatz dient der Unterstützung des Selbstmanagements eines Systems beim Auftreten von Ressourcenfehlern. Systematisch lässt sich das Vorgehen als Instanz des *MAPE-K*-Kreislaufes, vergleiche Abschnitt 2.2.5, beschreiben. Die Abbildung 5.1 ordnet die Modelle und Prozessaktivitäten des Ansatzes in den Kreislauf ein.

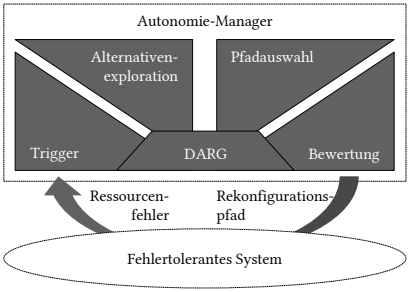


Abbildung 5.1.: Instanz des MAPE-K Regelkreises (Quelle: [KC03], angepasst)

Die Regelung bezieht sich auf das fehlertolerante System mit einem variantenreichen Konfigurationsraum als verwaltetes Element. Der deterministische Architekturrelationsgraph bildet die Wissensbasis (Englisch: knowledge) des

Entscheidungsprozesses. Ausgehend von Beobachtungen (Englisch: monitor) des Systems werden relevante Fehler innerhalb der aktuell gewählten Konfiguration lokalisiert (Betriebsphase) beziehungsweise synthetisch injiziert (Entwurfsphase). Die nachfolgende Analyse (Englisch: analyse) löst die Exploration innerhalb der Wissensbasis aus, ermittelt die Auswirkungen des Fehlers und erhebt mögliche Gegenmaßnahmen. Die Planung (Englisch: plan) greift die Informationen der Analyse auf und definiert einen möglichen Rekonfigurationspfad zu einer Alternativkonfiguration. Die Rekonfiguration erfolgt im letzten Schritt (Englisch: excute) entweder autark oder wird dem Anwender in geeigneter Form vorgeschlagen und argumentiert. Der ermittelte Rekonfigurationspfad beschreibt die Modifikationen am laufenden System.

5.3.2. Anwendung und Wartungsaufwände

Der Kreislauf lässt zu jeder Zeit Interaktionen mit den Entwicklern oder dem Wartungspersonal des Systems zu. Entsprechend des gewünschten Grades an Autonomie, ist das Expertensystem einerseits als Analysewerkzeug in der *Offline-Wartung* außerhalb des Systems nutzbar. Anderseits ist der Einsatz zur *Online-Wartung* als Teil der Systemimplementierung mit optionaler Erweiterung um Regeln zur selbstständigen Rekonfiguration möglich. Beide Wartungsausprägungen lassen sich in der Entwurfszeit (simulierter Betrieb) oder Laufzeit (realer Betrieb) einsetzen. In der Simulation werden synthetische Ressourcenfehler gezielt injiziert. Im Betrieb treten Fehler unkontrolliert auf und werden durch eine Fehlerlokalisierung auf einzelne Ressourcen eingegrenzt.

Der Wartungsaufwand lässt sich insbesondere im Bezug auf den Interaktionsbedarf mit dem Anwender untersuchen. Die Ausprägung der Veränderungen werden hier nicht in expliziten Konfigurationen dargestellt, sondern ausschließlich in deren Umfang, das heißt in zeitlichem Wartungsaufwand und Kommunikationsbedarf. Die skizzierten Verläufe sind als Annäherungen zu interpretieren. Die tatsächlichen Wartungszeiten hängen stark von den Auswirkungen und der Antizipierbarkeit eines Fehlers ab.

Die Ausbreitung der Graphen in x-Richtung beschreibt die Zeit, in denen sich das System im Wartungszustand befindet. Der Ausschlag auf der y-Achse beschreibt den Kommunikationsbedarf mit dem Anwender. Dieser

kann sich in der einfachen Mitteilung von Änderungsinformationen (*Info*) ausprägen oder in einem ausführlichen Statusbericht (*Diagnose*). Die größten Aufwände resultieren in einer interaktiven Einbeziehung des Anwenders als Entscheidungsträger in Verbindung mit manuellen Wartungstätigkeiten (*Implementierung*). Nachfolgend werden die möglichen Wartungsausprägungen im Kontext der Abbildung 5.2 erläutert.

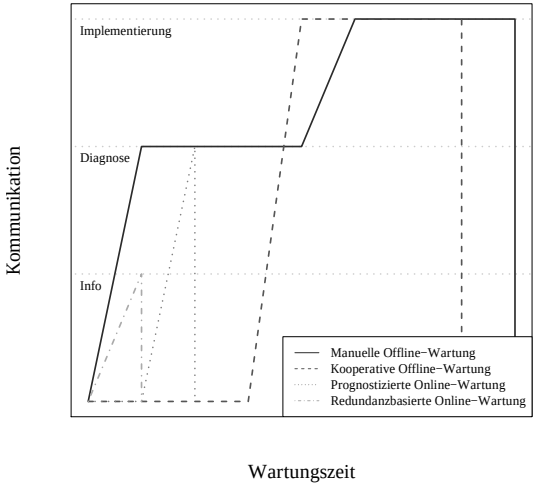


Abbildung 5.2.: Beteiligung des Anwenders im Entwicklungs-/Wartungsprozess

Redundanzbasierte Online-Wartung: Unter der Annahme, dass ein System fehlertolerant entworfen wurde, bestehen eine Vielzahl von symmetrisch strukturierten Redundanzbeziehungen. Diese Redundanzen führen zu strukturell minimalen Änderungen und lassen sich daher als Teil der Online-Wartung autark durchführen. Es ist strategisch naheliegend, dass im Entwicklungsprozess für alle Konstellationen der redundant verbauten Hardwareressourcen entsprechende logische Vor- und Nachverarbeitungen entwickelt werden und Teil der Betriebssoftware sind. Der Anwender wird im Falle eines Redundanzwechsels informiert, jedoch nicht am Entscheidungsprozess beteiligt.

Prognostizierte Online-Wartung: Ermittelt die Fehlerbehandlung selbstständig eine Änderung, die über einen Redundanzwechsel hinausgeht, steigt der Umfang von Rekonfiguration und Interaktion grundsätzlich. Handelt es sich jedoch um einen erwarteten Fehler, wird in der Entwurfsphase entsprechend eine alternative Implementierung vorbereitet. Durch die strukturell komplexen Änderungen im System sind ebenfalls die qualitativen Auswirkungen potenziell signifikant und eine umfangreiche Rückmeldung mit relevanten Diagnosedaten notwendig. Der Anwender wird so zwar über die Änderung informiert, muss diese aber nicht nachträglich implementieren. Somit wechselt das System schnell wieder in den operativen Zustand. Diese Art von Änderungen bedarf einen gesteigerten Aufwand in der Entwicklungsphase, da eine Vielzahl von alternativen Systemkonfigurationen implementiert und getestet werden müssen.

Manuelle Offline-Wartung: War der auftretende Fehler nicht prognostizierbar, erhält der Anwender eine Fehlermeldung vom System und muss die Wartung auf Basis dieser Daten manuell durchführen. Im Idealfall erkennt der Anwender aus den Diagnosedaten um welchen Fehler es sich handelt und kann gezielt eine Gegenmaßnahme ergreifen. Im schlechtesten Fall kommt es allerdings zu einer langen Ausfallzeit des Systems, da eine ausführliche manuelle Analyse der Daten durchgeführt werden muss und erst anschließend die Implementierung erfolgt.

Kooperative Offline-Wartung: Besitzt das System erweiterte Fähigkeiten, um die Auswirkungsanalyse eines Fehlers selbständig durchzuführen, kann die Kontaktaufnahme mit dem Anwender zeitlich verzögert und auf die wesentlichen Informationen reduziert werden. Eine aufgearbeitete Datenbasis fördert den effizienten Wartungsprozess. Die manuelle Implementierung der vorgeschlagenen Lösung bleibt weiterhin notwendig. Allerdings wird die vorgelagerte Analysephase dabei automatisiert und stark verkürzt.

Der vorgestellte Ansatz adressiert vorrangig eine Anwendung in der unterstützten Offline-Wartung, ist jedoch auch als Basis für einen autarken Entscheidungsmechanismus in der Online-Wartung denkbar. Weiterhin ist eine Anwendung in der früheren Phase der Systementwicklung zur Identifikation von Rekonfigurationsalternativen und deren vollständiger Implementierung möglich. Im weiteren Verlauf dieser Arbeit ist diese Anwendung in

der Wartung fokussiert. Dabei wird angenommen, dass qualitative Anforderungen frühzeitig an jede Systemkonfiguration gestellt und approximiert werden können. Hierbei ist weiterhin eine Konkretisierung und Auflösung von Zielkonflikten zwischen den Qualitätsanforderungen notwendig.

5.3.3. Auflösung qualitativer Zielkonflikte

In der multi-kriteriellen Optimierung wird grundsätzlich initial davon ausgegangen, dass sämtliche Kriterien von gleicher Relevanz für das Endergebnis sind. Dies findet sich insbesondere in der Definition der Pareto-Effizienz wieder, welche immer das Supremum von einem Kriterium fordern und keine globale Optimalität über alle Kriterien, vergleiche Abschnitt 2.2.3.

In einer realistischen Ausführungsumgebung eines Systems lassen sich Kriterien jedoch oftmals auf Grundlage von beispielsweise Leistungsanforderungen oder Erfahrungswerten grob priorisieren. Im Bereich der eingebetteten Systeme werden so für unterschiedliche Betriebsmodi spezifische Qualitätsanforderungen definiert. Die Abbildung Abbildung 5.3 zeigt eine exemplarische Pareto-Kurve und eine zusätzliche Gerade, die die Optimalität beider Qualitätsziele darstellt. Die notwendige Festlegung von *Trade-Offs* wird in Gewichtungen für die untersuchten Qualitätsattribute durch den Anwender bestimmt. Die Gerade repräsentiert entsprechend eine Gewichtung von 60% in Qualitätsdimension x zu 40% in Dimension y. Daraus resultiert die exemplarische Funktion

$$U(x, y) = 0,6 * x + 0,4 * y$$

zur Berechnung des Nutzwertes einer Systemkonfiguration. Zusätzlich werden minimale Akzeptanzwerte als Nebenbedingung der Funktion definiert, die die untersuchten Konfigurationen einhalten müssen. Dabei muss

$$x \geq 0.2 \wedge y \geq 0.1$$

für die jeweils erhobenen Wertebelegungen für x und y gelten. Entsprechend der Abbildung 5.3 werden C_0 und C_1 als die einzigen Pareto-effizienten Systemkonfigurationen angenommen. Es ergeben sich als Nutzwerte $U_{C_0} = 0,32$ und $U_{C_1} = 0,40$. Die Systemkonfiguration C_0 gilt somit nach der Analyse der *Trade-Offs* als *betriebsoptimal* für die gesetzten Qualitätsziele.

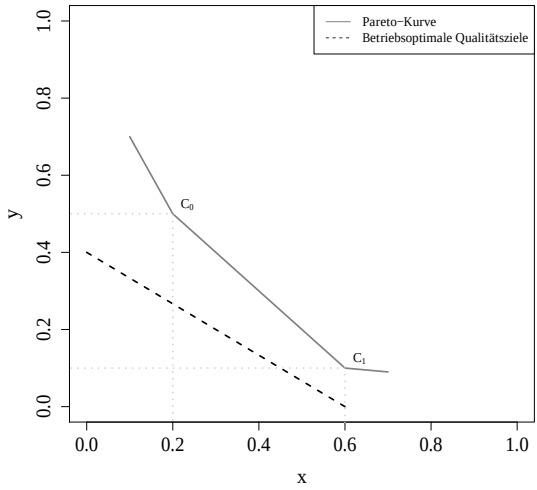


Abbildung 5.3.: Qualitätsziele im Pareto-Raum

Die mehrdimensionale Definition von Gewichtungen und minimalen Akzeptanzwerten sowie die Auswertung für gegebene Werte, werden im Ansatz innerhalb von Instanzen des Qualitätsattributsgraphens durchgeführt und anschließend in der Determinisierung des Architekturrelationsgraphens aufgegriffen.

In der Validierung in Abschnitt 6 wird von einem kooperativen Verbund einer umfangreichen Diagnose innerhalb des produktiven Systems und jeweils einer synchronisierten *DARG*-Modellinstanz pro Betriebsmodus im Bodensegment ausgegangen. Beides unterstützt das Wartungspersonal in der Beobachtung und dem Abwägen von Entscheidungen zur Identifikation und Umsetzung vorgeschlagener Aktivitäten.

Werden im Betrieb des Systems unterschiedliche Betriebsmodi vorgesehen, vergleiche Abschnitt 5.4, und ist eine frühzeitige zeitliche Anordnung dieser möglich, kann eine zusätzliche Optimierung zur Entwurfszeit stattfinden. Dies wird zusätzlich in Abschnitt 6.8 validiert.

5.4. Behandlung variierender Betriebsmodi

Unter Berücksichtigung einer Menge von Betriebsmodi ermöglicht der Ansatz eine Optimierung der Übergänge zwischen *DARG*-Modellinstanzen unter Beachtung minimaler Rekonfigurationskosten [MFKR18]. Dabei werden zudem optimale Reihenfolgen der Modi abgeleitet. Abweichend von dem bisherigen Vorgehen, werden dabei graphübergreifende Beziehungen von *DARG*-Instanzen für die zugrundeliegenden Betriebsmodi zusätzlich betrachtet.

5.4.1. Angleichung von variierenden Betriebsmodi und assoziierten Rekonfigurationsräumen

Betriebsmodi können im Lebenszyklus des Systems variieren. Dabei existieren Beziehungen zwischen *DARG*-Instanzen, die im Rekonfigurationsprozess zu berücksichtigen sind. Die Ableitung optimierter Reihenfolgen der Modi ist in Abbildung 5.4 skizziert.

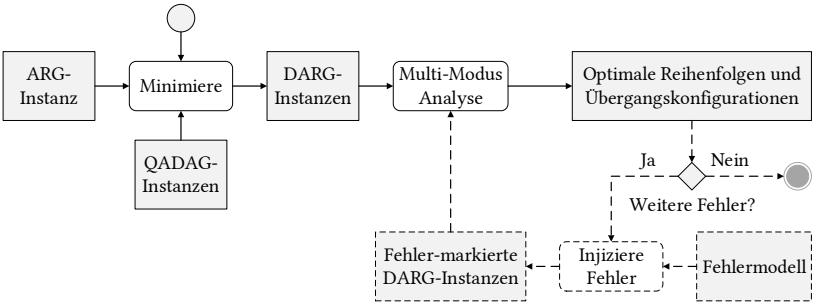


Abbildung 5.4.: Rekonfigurationsprozess mit variierenden Benutzungsmodi

Unter Betrachtung mehrerer *QADAG*- und *DARG*-Instanzen findet eine graphübergreifende Analyse statt. Zunächst wird auf Basis der zuvor eingeführten Methoden eine graphinterne Analyse durchgeführt. Dabei wird eine

QADAG-Vorlage für jeden Betriebsmodus parametrisiert und eine *DARG*-Instanz abgeleitet und optimiert. Für jede dieser Instanzen wird in der Multi-Modusanalyse die Relevanz jeder Konfiguration innerhalb ihres Ursprungsgraphens bestimmt. Aus den gewonnen Daten werden mögliche Konfigurationen für den Übergang zwischen Betriebsmodi abgeleitet und darauf aufbauend eine optimale Reihenfolge von Betriebsmodi. Zur Bestimmung der Robustheit der Sequenzen für ein fehlertolerantes System werden Fehler injiziert und Auswirkungen auf die Transitionskonfigurationen betrachtet.

Zur Beurteilung von Gemeinsamkeiten zwischen Instanzen eines deterministischen Architekturrelationsgraphens sind quantitative Eigenschaften zur Unterscheidung zu erheben.

5.4.2. Priorisierung von Konfigurationen beim Wechsel eines Betriebsmodus

Jede Konfiguration innerhalb einer *DARG*-Instanz wird bezüglich ihrer Adäquatheit zum Betriebsmoduswechsel bewertet. Zusätzlich zu den vorhandenen Kennzahlen für eine Konfiguration aus den vorherigen Analysen, werden Graphzentralitäten genutzt, um die Relevanz jedes Knotens zu bewerten.

Die Berechnung der Zentralitäten hat generell eine Komplexität von $O(n^2)$, hängt jedoch massiv von der Implementierung der Variante des Zentralitätsbegriffs ab. In Abhängigkeit der Größe des *DARG* und der verfügbaren Rechenleistung, kann zwischen drei wesentlichen Detaillierungsgraden gewählt werden. Der Ansatz sieht optional die Verwendung der ressourcensparenden Gradzentralität [Die12], der ausgewogenen pfadorientierten Nähenzentralität [Bav50] oder der ressourcenintensiven graphübergreifenden Zwischenzentralität [Fre77] vor.

Insgesamt resultierten drei *Transitionskriterien* zur Bewertung der graphübergreifenden Rekonfigurierbarkeit jeder Konfiguration. Aus Abschnitt 4.1.1 werden der normalisierte Nutzwert $utility$ und die Menge der Ressourcenbindungen *bind* pro Konfiguration aufgegriffen. Ersterer Wert definiert die Adäquatheit der Konfiguration zu dem aktuellen Betriebsmodus. Die Ressourcenbindungen werden beschreiben den Grad an Redundanz. Zusätzlich wird die Zentralität *cen* des zugehörigen Knotens einer Konfiguration als drittes Kriterium aufgegriffen. Die Zentralität quantisiert in diesem Kontext die

Rekonfigurationsoptionen. Zur Wertkumulation werden die Ressourcenbindungen und die Zentralitäten über alle auftretende Werte in der betrachteten *DARG*-Instanz normalisiert. Der Anwender kann durch eine Vergabe von Gewichtungen $w_1, w_2, w_3 \in [0; 1]$ die Relevanz einzelner Kriterien im Modusübergang festlegen. Die Gewichte sind konstant über alle *DARG*-Instanzen festzulegen.

Durch Kombination aller Werte ergibt sich für jede Konfiguration c_i ein *Transitionswert* in Bezug auf deren Ursprungsgraphen (D_i), definiert als:

$$tv_{c_i,D_i} := w_1 * utility_{c_i} + w_2 * cen_{c_i} + w_3 * |bind_{c_i}|$$

5.4.3. Ableitung optimaler Sequenzen für Betriebsmodi

Zur Ableitung optimaler Sequenzen für Betriebsmodi, greift unser Ansatz die Transitionswerte ähnlicher Konfigurationen innerhalb von Paaren von *DARG*-Instanzen auf. Durch den gemeinsamen *ARG*, besteht eine hohe Wahrscheinlichkeit zur Identifikation von ähnlichen Konfigurationen. Die Identität einer Konfiguration wird über einen Vergleich des Identifikators festgestellt und wie folgt verglichen: Seien $\mathcal{D}$ die Menge aller *DARG*-Instanzen und D_s und D_t ein Paar in $\mathcal{D}$, dann ist $D_s \cap_{id} D_t \iff \forall c_i \in D_s, c_j \in D_t : c_{i_{id}} == c_{j_{id}}$ der Schnitt der gemeinsamen Konfigurationen des Paars.

Auch wenn es unwahrscheinlich ist, dass die Schnittmenge leer ist, wäre eine Bildung einer Reihenfolge des Paars nicht möglich. Wir betrachten diesen Aspekt zum Ende dieses Abschnitts.

Zur Untersuchung der Transitionswerte zwischen den Paaren, werden die Werte aufsummiert. Durch die Asymmetrien in den Wertberechnungen je nach Ursprungsgraphen, werden zwei Werte pro Paar berücksichtigt. Exemplarisch ergibt sich für das Paar D_s und D_t folgende *Transitionswertsumme*:

$$tvs_{D_s \cap_{id} D_t} := \left(\sum_{i=1}^{|D_s \cap_{id} D_t|} tv_{c_i,D_s} + tv_{c_i,D_t} \right) \mid c_i \in D_s \cap_{id} D_t$$

Zur Berücksichtigung aller möglichen Paare wird eine teilweise Permutation über $\mathcal{D}$ durchgeführt. Dabei ist $S \subset \mathcal{P}(\mathcal{D})$ mit $S_i \in S$ eine *geordnete*

Betriebsmodussequenz wobei die Größe jeder Permutation der Menge aller *DARG*-Instanzen entspricht: $|S_i| == |\mathcal{D}|$.

Das Maximum aller Transitionswertsumme in S wird abgeleitet durch:

$$tvs_{max} := \max \sum_{i=1}^{|S|} \left(tvs_{D_s \cap_{id} D_t} \forall D_s, D_t \in S_i \right) \mid S_i \in S$$

Die Sequenz mit der höchsten Transitionswertsumme entspricht der *optimalen Sequenz* der *DARG*-Instanzen. Jede Konfiguration in dieser Sequenz ist eine *Transitionskonfiguration.*

Sind Schnittmengen zwischen Graphpaaren leer, gilt $tvs = 0$. Dadurch wird das Paar im Zuge der Maximierung abgewertet. Dieser Umstand hat nur dann eine Auswirkung auf die totale Ordnung aller *DARG*-Instanzen, wenn das Paar direkt nebeneinander angeordnet ist. In diesem Fall wird die Reihenfolge zufällig festgelegt, damit insgesamt zumindest eine partielle Ordnung ableitbar ist.

Durch die Zurückverfolgung jeder *DARG*-Instanz zu einer *QADAG*-Instanz, wird der zugehörige Betriebsmodus ermittelt. Folglich ergibt sich im Ergebnis der Analyse die *optimale Sequenz der Betriebsmodi.*

Die Ableitung der *DARG*-Instanzen für mehrere Betriebsmodi erfolgt mit einer Komplexität von $O(n)$. Somit liegt die Ermittlung und Abwägung von Transistionskonfigurationen für alle Paare in $O(n^2)$.

5.4.4. Berücksichtigung von Ressourcenfehlern

Jeder Ressourcenfehler reduziert die Menge ausführbarer Konfigurationen. Dies führt insbesondere dazu, dass Konfigurationen vollständig entfallen können. Dies hat Auswirkungen auf die Gemeinsamkeiten zwischen *QADAG*-Instanzen, der allgemeinen Ressourcenverfügbarkeit und die Zentralitäten innerhalb der Graphen. Ressourcenfehler beeinflussen somit die initiale Auswahl von Transitionskonfigurationen und die Festlegung optimaler Modussequenzen signifikant. Zur Berücksichtigung dieses Aspekts betrachten wir Fehlerauswirkungen ebenfalls in dieser Analyse frühzeitig in der Entwurfsphase.

Die Stabilität der Transitionskonfigurationen und Modussequenzen wird nach jedem injizierten Fehler in einem iterativen Prozess beurteilt. Zunächst wird die betrachtete Ressource als fehlerhaft in der Ressourcenplattform markiert. Anschließend werden alle Paare der *DARG*-Instanzen in S_i auf deren Ausfallintensität, vergleiche Definition 4.5 in Abschnitt 4.1.1, untersucht. Sobald ein Paar betroffen ist wird es zur Neuordnung markiert und alle Transitionswerte werden neuberechnet. Nachdem alle Instanzen abgearbeitet sind, wird die Reihenfolge der Modussequenz entsprechend der aktualisierten maximalen Transitionssummen aktualisiert.

Nachdem alle Fehler injiziert wurden, wird die letzte Reihenfolge als neue optimale Sequenz der Betriebsmodi unter Betrachtung des schlimmstmöglichen Fehlerfalls festgehalten. Eine Sequenz ist stabil, wenn sie keinen Änderungen unterliegt.

6. Validierung

Zur Beurteilung der Effektivität des Ansatzes zur effizienten Entscheidungsunterstützung in der Wartung, untersucht die Validierung die Verringerung der Aufwände in der Kommunikation und den Tätigkeiten des Wartungspersonals. Die Fallstudie hat den Entwicklungs- und Wartungsprozess des Lageregelungssystems eines Kleinsatellitens zum Gegenstand.

6.1. Überblick

Der entwickelte Ansatz soll die Kommunikationsaufwände zwischen System und Anwender reduzieren und zugleich die Wartungstätigkeiten unterstützen. Die Selbstwartung soll weiterhin konservativ bezüglich des Grades an Autonomie sein und vorrangig als Expertensystem für eine transparente Entscheidungsfindung dienen. Entsprechend wird der Ansatz in dieser Arbeit als Hilfsmittel in der Offline-Wartung, vergleiche Abschnitt 5.3.2, erprobt und beurteilt.

6.1.1. Vorgehen

Die Validierung ist in zwei Teilaspekte untergliedert. Zum einen wird eine empirische Untersuchung mit Domänenexperten ausgearbeitet und durchgeführt, zum anderen erfolgt eine werkzeuggestützte Simulation des Wartungsbetriebs des Anwendungsszenarios. Zur Erhebung der Ist-Situation in der traditionellen Wartung von fehlertoleranten Systemen, wird ein strukturiertes Experteninterview mit Ingenieuren aus der Raumfahrtsystementwicklung durchgeführt. Im Rahmen der Befragung werden Entwurfsentscheidungen und betriebliche Wartungsaufwände ermittelt und die Gültigkeit der Annahmen des entwickelten Ansatzes überprüft. Die simulativen Messungen

erfolgen auf Basis einer erweiterten Variante des Lageregelungssystems mit einem erhöhten Grad an Redundanz und Variabilität in Sensorik und Aktuatorik. Dabei wird ein variantenreicher Entwurf des Systems modelliert und anhand des Ansatzes auf Konfigurierbarkeit untersucht. Zur Beurteilung der Effektivität und Effizienz des Ansatzes, werden für exemplarische Fehlerszenarien die Beschaffenheit möglicher Entwurfsräume beurteilt. Dabei werden verschiedene Minimierungen des Rekonfigurationsraums und Fehlerintensitäten der Ressourcenplattform betrachtet.

6.1.2. Validierungsaufbau nach GQM

Die Validierung orientiert sich nach dem GQM-Modell (Englisch: Goal-Question-Metric) von Basili und Weiß [BW84]. In sechs Stufen werden neben den Zielsetzungen der Aufbau und die Durchführung der Messungen beschrieben. Dabei werden (1.) der allgemeine Kontext der Untersuchung festgelegt, (2.) Fragen zur Beurteilung der Ziele formuliert und (3.) Metriken zur Beantwortung der Fragen aufgestellt. Die Metriken adressieren in dieser Arbeit einerseits quantitative werkzeuggestützte und andererseits qualitative empirische Messungen. Weiterhin werden (4.) die Messungen in geeigneter Form durchgeführt, (5.) eine Auswertung sowie Interpretation der Messdaten vorgenommen und abschließend (6.) die aufgearbeiteten Daten zur Beurteilung der Zielerreichung auf die Fragen zurückgeführt.

In der ersten GQM-Stufe (kurz: *GQM1*) wird der Untersuchungskontext eingegrenzt. In dieser Arbeit werden dazu die Forschungsfragen aus Abschnitt 1.2.3 rekapituliert, vergleiche Abschnitt 6.2. Darauf aufbauend werden, zur exakten Definition der Validierungsziele in *GQM2*, aus den allgemeinen Forschungsfragen feingranulare Validierungsfragen formuliert, vergleiche Abschnitt 6.3. Im gleichen Abschnitt werden entsprechend *GQM3* Metriken identifiziert. Diese Metriken werden zur späteren Beurteilung der Zielerreichung der Validierungsfragen herangezogen. Dabei wird auf das Anwendungsszenario, vergleiche Abschnitt 6.4, Bezug genommen. Die konkreten Definitionen der Metriken in *GQM4* beziehen sich teilweise auf den empirischen Teil der Validierung, vergleiche Abschnitt 6.5, sowie auf die werkzeuggestüzte Analyse, vergleiche Abschnitt 6.6. Die Durchführung der Messungen nach *GQM5* erfolgt in vier Schritten. Zunächst werden die (1)

gesuchten Daten klassifiziert und (2) deren Abhängigkeiten erhoben. Den wesentlichen Teil der Messung macht die (3) Datenerhebung aus und die darauf aufbauende (4) Aufbereitung der Messergebnisse. Die Auswertung sämtlicher Validierungsdaten gemäß *GQM6* erfolgt in Abschnitt 6.7 im Vergleich zum Status Quo der bekannten Wartungsaktivitäten. Zunächst wird individuell Bezug auf die Beantwortung der Validierungsfragen genommen. Im Anschluss erfolgt in Abschnitt 6.9 die Gesamtbeurteilung der Zielerreichung der Validierung im Bezug auf die Forschungsfragen sowie eine Diskussion von Limitierungen und Erweiterungen.

6.2. Eingrenzung der Untersuchung

Zur Untersuchung der allgemeinen Forschungsfragen dieser Arbeit aus Abschnitt 1.2.3 sind gemäß *GQM1* die Kernaspekte herauszuarbeiten und in messbare Validierungsfragen zu überführen. Diese Valdierungsfragen beschreiben die Zielsetzungen der Validierung.

FF1 *„Welche Faktoren limitieren die Voraussagekraft der Konfigurierbarkeit eines Systems unter Annahme kontextfreier Fehlerszenarien, festgelegter Betriebsmodi und statischer Qualitätseigenschaften?“*

Forschungsfrage **FF1** erhebt die Faktoren und Effekte, die Prognosen über die Konfigurierbarkeit eines Systems beeinflussen. Dabei wird angenommen, dass Betriebsszenarien und qualitative Eigenschaften bekannt sind und Fehlerauswirkungen ohne Beachtung des Systemkontextes bewertet werden können.

FF2 *„Lassen sich Fernwartungsaufwände im Betrieb durch approximative Priorisierungen von Rekonfigurationsentscheidungen zur Entwurfszeit signifikant reduzieren?“*

In Forschungsfrage **FF2** steht die Investition von Mehraufwänden in der Entwicklung zur Reduktion von Wartungskosten im Vordergrund. Die Aufwände beziehen sich auf die Ermittlung einer Entscheidungspriorisierung zur signifikanten Vereinfachung des Wartungsprozesses.

FF3 *„Lässt sich die Prozesstransparenz in der Fernwartung durch den Einsatz komponentenbasierter Modelle effektiv unterstützen?“*

Die Transparenz im Prozess der Fernwartung wird von Forschungsfrage **FF3** behandelt. Die Validierung soll dabei die Effektivität des Einsatzes von komponentenbasierten Modellen überprüfen.

FF4 *„Liefern meta-heuristische Suchverfahren auf Architekturbasis eine adäquate Entscheidungsbasis zur Festlegung weitreichender, nicht trivial identifizierbarer, Systemkonfigurationen mit qualitativen Abstufungen?“*

Der Einsatz eines meta-heuristischen Suchverfahrens auf Architekturbasis wird in Forschungsfrage **FF4** adressiert. Im Kern der Frage steht die Vollständigkeit der erhobenen Wissensbasis und die Priorisierung von Alternativkonfigurationen nach qualitativen Eigenschaften.

6.3. Erhebung von Validierungsfragen und Identifikation von Metriken

Zur Präzisierung der Zielsetzung werden gemäß *GQM2*, vergleiche Abschnitt 6.1.2, messbare Validierungsfragen aufgestellt. Als Datenquellen liegen die Aufzeichnungen aus den Experteninterviews vor, sowie die Analyseergebnisse aus der technischen Umsetzung des Ansatzes. Auf dieser Basis werden Metriken entsprechend *GQM3* identifiziert, die nachfolgend Maßnahmen zur Erfassung und Aufbereitung der Rohdaten in qualitativen und quantitativen Messungen beschreiben. Es wird zunächst ein Überblick über die identifizierten Schwerpunkte der Validierung gegeben. Qualitative Auswertungen werden im Zuge der Interviews in Abschnitt 6.5.3 ausformuliert. Die Umsetzung quantitativen werkzeuggestützten Metriken sind im Kern von Abschnitt 6.6.4.

6.3.1. Verfeinerung der Forschungsfrage FF1

FF1 *„Welche Faktoren limitieren die Voraussagekraft der Konfigurierbarkeit eines Systems unter Annahme kontextfreier Fehlerszenarien, festgelegter Betriebsmodi und statischer Qualitätseigenschaften?“*

Im Einzelnen sind im Rahmen der Forschungsfrage **FF1** folgende Validierungsfragen zu beantworten:

FF1.1 *„Ist eine Abstraktion des Laufzeitkontextes im realen Systementwurf legitim?“*

FF1.2 *„Ist der Systemzustand nach Fehlereintritt eindeutig beschreibbar?“*

FF1.3 *„In welchem Umfang werden Fehlerauswirkungsanalysen durchgeführt?“*

FF1.4 *„Ist eine statische Betrachtung von Qualitätseigenschaften ausreichend?“*

Zur Beurteilung der Zielerreichung aller vier Verfeinerungen der Forschungsfrage **FF1** werden die Experteninterviews empirisch ausgewertet. Neben den Erfahrungen aus der Entwicklung (**FF1.1**) und Wartung (**FF1.2**) von fehlertoleranten Raumfahrtsystemen, wird die Verfügbarkeit von Wissen aus Analysemodellen (**FF1.3**), Datenblättern und historischen Daten von Vorläufersystemen (**FF1.4**) näher untersucht.

6.3.2. Metrikenidentifikation zu Validierungsfragen FF1.1 bis FF1.4

Um **FF1.1** positiv zu bewerten, ist zu belegen, dass im existierenden Entwicklungsprozess bereits Prognosedaten genutzt werden, die hinreichend unabhängig vom Ausführungskontext sind. Die Frage untersucht, ob eine Einbeziehung von Umweltmodellen zwingend notwendig ist.

Die Erkenntnisse aus dem laufenden Wartungsprozess sollen **FF1.2** beantworten. Dabei gilt es zu zeigen, dass der Systemzustand jederzeit im Betrieb eindeutig bestimmbar ist und sich dieser mit den Prognosen aus der Entwicklung deckt.

Ebenfalls mit Bezug auf den Wartungsprozess wird nach Antworten auf **FF1.3** gesucht. Diese Frage richtet sich nach der produktiven Anwendbarkeit von frühzeitigen Fehlerauswirkungsanalysen. Eine gültige Beantwortung beinhaltet die Bestätigung, dass entsprechende Ansätze akzeptiert sind.

In **FF1.4** soll insbesondere die Relevanz von statischen Informationen aus Datenblättern von Bauteilen beurteilt werden. Zur Beantwortung wird nach

Aussagen gesucht, die Datenblätter als wichtige Quelle für Prognosen auszeichnen. Ebenfalls werden die Rolle historischer Daten aus vorausgehenden Missionen in die Betrachtung einbezogen.

6.3.3. Verfeinerung der Forschungsfrage FF2

FF2 *„Lassen sich Fernwartungsaufwände im Betrieb durch approximative Priorisierungen von Rekonfigurationsentscheidungen zur Entwurfszeit signifikant reduzieren?“*

Im Details sind die nachfolgenden Fragen zur Validierung von Forschungsfrage **FF2** zu untersuchen:

FF2.1 *„Hängen personelle Aufwände primär von Diagnose- oder Reparaturtätigkeiten ab?“*

FF2.2 *„Resultiert der wesentliche Kommunikationsaufwand aus Datenmengen oder Latenzen im Systemzugriff?“*

FF2.3 *„Lässt sich anhand vorliegender Diagnosedaten die verbleibende Konfigurierbarkeit des Systems nachvollziehen?“*

FF2.4 *„Ist ein modellbasierter Konfigurationsbegriff aus der Entwurfsphase für die Wartungsanalyse zur Laufzeit einsetzbar?“*

FF2.5 *„Ist der Kommunikationsaufwand im Betrieb durch frühzeitige Analysen des Entwurfsraums reduzierbar?“*

FF2.6 *„Führt eine Priorisierung von Rekonfigurationsentscheidungen zu einer Reduktion des personellen Wartungsaufwands?“*

Die Forschungsfrage **FF2** wird einerseits anhand der Experteninterviews empirisch untersucht und andererseits anhand von Messdaten aus der werkzeuggestützten Analyse. So werden Aufwände in Personal (**FF2.1**) und Kommunikation (**FF2.2**) aus den Angaben der Experten präzisiert. Ergänzend werden die notwendigen Parameter zur Beurteilung des Systemzustandes (**FF2.3**) im Betrieb erhoben und eine Einordnung dieses Zustandes zu den Konfiguration (**FF2.4**) im Entwurf vorgenommen. Als Kernbeitrag der werkzeuggestützten Validierung, wird die Reduzierbarkeit der Aufwände in Kommunikation (**FF2.5**) und Personaleinsatz (**FF2.6**) ausführlich untersucht.

6.3.4. Metrikenidentifikation zu Validierungsfragen FF2.1 bis FF2.6

Zur Abgrenzung der personellen Wartungsaufwände soll in **FF2.1** beantwortet werden, ob die Tätigkeiten in der Diagnose ausschlaggebend sind oder ob der hauptsächliche Aufwand in der Reparatur vorliegt. Dabei wird für tatsächliche Wartungsfälle erhoben wie zeitaufwendig die Fehleranalyse und die Behebung (Englisch: *Time-to-Repair*) einer Lösung ist.

Ergänzend zur vorherigen Forschungsfrage, sucht **FF2.2** nach dem Ursprung der Kommunikationsaufwände. Die Befragung soll Aufschluss darüber geben, ob die Datenmengen oder der nicht-unterbrechungsfreie Systemzugriff die limitierenden Faktoren sind. Dazu werden Datenmengen, Schnittstellenkapazität und Verfügbarkeit des Systemzugriffs ermittelt.

FF2.3 überprüft, ob im aktuellen Wartungsprozess mit minimalem Kommunikationsaufwand der aktuelle Systemzustand identifizierbar ist. Zur Beantwortung der Frage wird der Umfang der aktuell übertragenden Diagnosedaten (Englisch: *Housekeeping Data*) erhoben und zur Repräsentation des Systemzustands beurteilt.

Die Antwort auf **FF2.4** revidiert, ob ein Bezug zwischen den Analysemodellen im Entwurf und dem Plattform-Quellcode möglich ist. Dazu wird erhoben welche Bedeutung Architekturen im Entwurfsprozess haben und ob diese Artefakte in der Wartung aufgegriffen werden.

Zur Beantwortung von **FF2.5** werden die Kennzahlen aus Abschnitt 5.2.4 aufgegriffen. Die Messung vergleicht den Kommunikationsaufwand zur Übertragung des Konfigurationspfades mit allen relevanten Informationen zu qualitativen Degradationen und Ressourcenänderungen mit einer üblichen Fehlermeldung.

Die Kennzahlen aus Abschnitt 5.2.4 unterstützen weiterhin die Messungen für **FF2.6**. Die Effizienz wird in Form der notwendigen Rekonfigurationsschritte gemessen. Die Effektivität wird in qualitativer und struktureller Stabilität bemessen.

6.3.5. Verfeinerung der Forschungsfrage FF3

FF3 *„Lässt sich die Prozesstransparenz in der Fernwartung durch den Einsatz komponentenbasierter Modelle effektiv unterstützen?“*

Die nachfolgenden Unterpunkte werden auf Basis von Forschungsfrage **FF3** überprüft:

FF3.1 *„Ist eine kompakte visuelle Darstellung des Rekonfigurationsraums umsetzbar?“*

FF3.2 *„Sind Zusammenhänge zwischen Fehlereintritt und Fehlerauswirkung nachvollziehbar darstellbar?“*

FF3.3 *„Ist der vorgestellte Ansatz in bestehende Wartungsprozesse zur Fehlerbehandlung integrierbar?“*

FF3.4 *„Ist eine vorrangig modellbasierte Dokumentation der Wartungsschritte zielführend?“*

Die Nachvollziehbarkeit und Akzeptanz des Prozesses entsprechend Forschungsfrage **FF3** wird in Bezug auf die werkzeuggestützte Modellierung untersucht. Zunächst wird die strukturelle Komplexität der erzeugten deterministischen Architekturrelationsgraphen (**FF3.1**) ohne Fehlerannahmen eruiert. Anschließend erfolgt eine Untersuchung der Rekonfigurationspfade als nachvollziehbarer Lösungsvorschlag nach einem Fehlereintritt (**FF3.2**). Wiederum empirisch eingestuft werden die Einbettung des Modellierungsverfahrens (**FF3.3**) in etablierte Konzepte zur Fehlerbehandlung und die Aussagekraft der Modelle (**FF3.4**) für die Wartung.

6.3.6. Metrikenidentifikation zu Validierungsfragen FF3.1 bis FF3.4

Die Darstellbarkeit des fehlerfreien Rekonfigurationsraums wird für **FF3.1** durch eine Messung der Komplexität des deterministischen Architekturrelationsgraphens der Fallstudie beurteilt. Die Komplexität wird durch Größe und Dichte des effektiven Entwurfsraums, vergleiche Abschnitt 5.2.4, charakterisiert. Die Kohäsion hängt kausal von den strukturellen Unterschieden der jeweils eingesetzten Ressourcen, ausgedrückt als Rekonfigurationskosten,

ab. Der Raum ist kompakt darstellbar, wenn eine deutliche Verbesserung der Vollvermaschung des Status quo erreicht wird.

Zur Messung nach **FF3.2** wird ein relevanter Fehler, der die aktuelle Konfiguration korrumpiert, in das System eingebracht und anschließend die Darstellung der Fehlerauswirkung beurteilt. Die Verständlichkeit wird als Anzahl transienter und wegfallender Konfigurationen von der aktuellen zur alternativen Konfiguration gemessen. So werden die Rekonfigurationsschritte gezählt, die erledigt werden müssen.

Die Integration des Ansatzes wird in **FF3.3** diskutiert. Eine exemplarische Einbettung des Ansatzes in mindestens ein Fehleranalyseverfahren ist zur Beantwortung der Frage plausibel vorzustellen.

Im Zuge der Erörterung von **FF3.4** wird die Angemessenheit des Abstraktionsgrades eines Rekonfigurationspfades, bezüglich der industriellen Anwendbarkeit in der Wartung, beurteilt. Konkret wird in dieser Frage im Dialog mit den Experten überprüft, ob sich rein architekturorientierte Änderungen realistisch auf Ebene des Plattform-Quellcodes zuordnen lassen.

6.3.7. Verfeinerung der Forschungsfrage FF4

FF4 *„Liefern meta-heuristische Suchverfahren auf Architekturbasis eine adäquate Entscheidungsbasis zur Festlegung weitreichender, nicht trivial identifizierbarer, Systemkonfigurationen mit qualitativen Abstufungen?“*

Zur Validierung von Forschungsfrage **FF4** werden die folgenden Aspekte näher betrachtet:

FF4.1 *„Ermittelt ein metaheuristisches Suchverfahren einen relevanten Ausschnitt des alternativen Entwurfsraums?“*

FF4.2 *„Lassen sich Differenzen von Konfigurationen auf rein struktureller und qualitativer Ebene adäquat beurteilen?“*

FF4.3 *„Treten qualitative Abstufungen zwischen adäquaten Entwurfsalternativen in einem relevanten Maße in realen Systemen auf?“*

Unter Betrachtung der Analyseergebnisse der werkzeuggestützten Simulation sowie der Expertenaussagen, wird eine generelle Beurteilung des Ansatzes entsprechend Forschungsfrage **FF4** vorgenommen. Neben der Anwendung von Metaheuristiken zur Erzeugung valider Konfigurationen (**FF4.1**), soll der Abstraktionsgrad der ausschließlich strukturbasierten Analyse (**FF4.2**) bewertet werden. Eine wesentliche Nebenbedingung ist dabei eine Messung der qualitativen Streuungen (**FF4.3**) im Lösungsraum.

6.3.8. Metrikenidentifikation zu Validierungsfragen FF4.1 bis FF4.3

Die Adäquatheit einer meta-heuristischen Untersuchung steht im Zentrum von **FF4.1**. Die Abdeckung eines relevanten Teils des möglichen Suchraums wird gemessen anhand der maximalen Längen möglicher Pfade im deterministischen Architekturrelationsgraph und den entstehenden Differenzen der eingesetzten Ressourcen. Hierdurch wird überprüft, ob eine Vielzahl unterschiedlicher Konfigurationen durch das *Sampling* erhoben wird.

FF4.2 ist im Rahmen einer empirischen Untersuchung zu beantworten. Die meta-heuristische Optimierung grenzt Konfigurationen rein durch qualitative Unterschiede voneinander ab. Die Ressourcenauswahl entsteht dabei implizit. Die Frage ist dadurch zu beantworten, ob andere Faktoren existieren, die eine qualitative Bewertung einer Konfiguration massiv beeinflussen, zum Beispiel Geschäftsentscheidungen.

Zur Beantwortung von **FF4.3** wird die Varianz der Belegungen aller Qualitätsdimensionen auf den längsten Pfaden im deterministischen Architekturrelationsgraphen untersucht. Die Frage wird positiv beantwortet, wenn relevante qualitative Degradationen erkennbar sind.

6.4. Anwendungsszenario Lageregelungssystem eines Kleinsatellitens

Zur Validierung des vorgestellten Ansatzes wird als Anwendungsszenario das Lageregelungssystem eines Satellitens herangezogen, der sich seit sechs Jahren im produktiven Betrieb befindet.

6.4.1. Kleinsatellit TET-1

In dieser Arbeit wird das Lageregelungssystem (Englisch: *Attitude Control System (ACS)* [LHSR12]) des Kleinsatellitens *TET-1* [FLK+10] behandelt. Der **T**echnologie**E**rprobungs**T**räger-**1** wurde im Rahmen des Programms *On-Orbit-Verifikation* (OOV) des *DLR*[1] initiiert und durch das *Bundesministerium für Wirtschaft und Technologie* gefördert.

Das Satellitensystem ist als kostengünstige Demonstrationsplattform für die Verifikation von experimenteller Hardware für den Weltraumeinsatz entworfen. Das DLR hat im Jahr 2008[2] den Auftrag zur Fertigung des Satellitens, Aufbau des Bodensegments und den Start an die *Kayser-Threde GmbH*[3] übergeben. In Unterauftrag wurde der TET-1 von der *Astro- und Feinwerktechnik Adlershof GmbH*[4] entworfen und integriert. Der Start des Systems erfolgte am 22. Juli 2012, der Beginn des produktiven Betriebs im Oktober des gleichen Jahres.

Die Abbildung 6.1 zeigt den TET-1 mit ein- beziehungsweise ausgestellten Solarpaneelen und Nutzlast. Die Außenmaße des Systems betragen in der dargestellten Position $88 \times 58 \times 67$ beziehungsweise $88 \times 154 \times 67$ cm in Höhe, Breite und Tiefe. Die Gesamtmasse liegt bei 120 kg inklusive 50 kg Nutzlast.

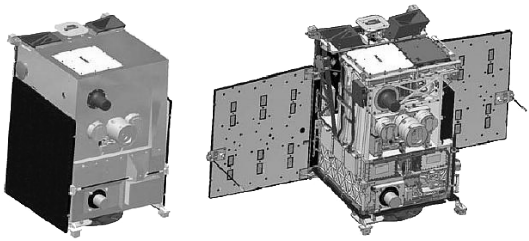


Abbildung 6.1.: TET-1 Kleinsatellit mit ein- und ausgeklappten Solarpaneelen (Quelle: Kayser-Threde GmbH, OHB System AG)

[1] Deutsches Zentrum für Luft- und Raumfahrt, `http://www.dlr.de`
[2] OOV TET-1 im Earth Observation Portal, `https://directory.eoportal.org/web/eoportal/satellite-missions/t/tet-1`
[3] Fusioniert als OHB System AG, `www.ohb-system.de`
[4] Industrieller Partner für das Anwendungsszenario, `http://www.astrofein.com`

6.4.2. Experimente und Missionsplanung

Das System vollzog im ersten Jahr seines Betriebs erfolgreich elf Experimente zur Technologieerprobung [AMS+10]. Dabei wurden einzelne wissenschaftliche und kommerzielle Baugruppen auf ihre Beständigkeit gegen kosmische Strahlung, Schwerelosigkeit und große Schwankungen von Hitze und Kälte in kurzen Perioden verifiziert. Die experimentelle Hardware wurde als Nutzlast nach den Entwicklungsvorgaben für den TET-1 entwickelt und nach Integration in Abnahmetests verifiziert. Die Experimente erstrecken sich über eine Vielzahl technologischer Bereiche, im Einzelnen:

- Drei unterschiedliche Solarzellen
- Eine Lithium-Polymer Batterie
- Zwei GPS-Empfangssysteme
- Ein Sensor-Datenbussystem
- Ein Antriebssystem für Picosatelliten
- Eine Infrarot-Kamera
- Ein Hauptrechner-System
- Ein RF-Kommunikationssystem

Für das erste Betriebsjahr wurden die Missionen über den gesamten Zeitraum langfristig geplant[SLW+14]. Eine Woche vor dem geplanten Missionsbeginn werden die Laufzeitdaten gesammelt und aufbereitet. Anschließend erfolgt eine Feinjustierung der Mission an den Zustand des ACS. Zwei Tage vor der Mission werden die Daten auf den TET-1 übertragen, wo anschließend die Ausführung erfolgt.

Nach Ablauf des OOV-Programms ging der Satellit in eine Folgemission über. Seither ist TET-1 Teil der FireBIRD-Mission[5]. Im Verbund mit seinem Nachfolger BIROS detektiert das System Waldbrände, Vulkanausbrüche und weitere Hochtemperaturvorfälle auf der Erde. Das System hat eine Umlaufzeit von 95 Minuten und fliegt im niedrigen Erdorbit in einer Höhe zwischen 450 und 850 km (niedriger Erdorbit).

[5] FireBIRD-Mission, `http://www.dlr.de/dlr/desktopdefault.aspx/tabid-10885/`

6.4.3. Lageregelung

Insbesondere die Lageregelung eines Satelliten erweist sich als kritisch für den Betrieb. Studien [RS03] zufolge treten ein Drittel aller Anomalien in diesem Subsystem auf und 40% aller Fälle führen aufgrund von Software- oder Hardwareausfällen zum Verlust des Satellitens [LHSR12]. Dies bedingt hohe Anforderungen an die Zuverlässigkeit und somit ein inhärent hohes Maß an Fehlertoleranz. Zur Untersuchung des Rekonfigurationsraums eines solchen Systems, wird die Lageregelung, vergleiche [MC14], des TET-1 als Anwendungsszenario dieser Arbeit herangezogen.

Das ACS [YTRM10] (Englisch: Attitude Control System[6]) bestimmt und ändert die Ausrichtung und Positionierung des Satellitens. Bei dem Subsystem handelt es sich um eine Weiterentwicklung der flugerprobten Umsetzung des BIRD-Mikrosatelliten [MD08].

Weiterhin erfordern die elf Experimente unterschiedliche Lageregelungsmodi, zum Beispiel eine Sonnenausrichtung oder eine Zielausrichtung auf einen definierten Punkt auf der Erde. Zur Erfüllung beider Anforderungen ist das System mit einer Vielzahl von Sensoren und Aktuatoren ausgestattet.

In der Abbildung 6.2 sind der schematische Aufbau des Softwaresystems und die eingesetzten Ressourcen (Bauteilgruppen) der Lageregelung des TET-1 skizziert. Die Ressourcen lassen sich in die folgenden zwei Aktuatoriksysteme und fünf Sensoriksysteme aufteilen:

- Vier heißredundante Reaktionsräder
  (RW, Englisch: Reaction Wheel)
- Zwei kaltredundante Magnetspulen pro Achse im Koordinatensystem
  (MC, Englisch: Magnetic Coil)
- Zwei kaltredundante Magnetfeldsensoren
  (MFS, Englisch: Magnetic Field Sensor)
- Zwei heißredundante Sonnensensoren mit je zwei Sensoren auf der Vorder- und Rückseite des TET-1
  (CSS, Englisch: Coarse Sun Sensor)

[6] Teilweise abweichend auch als Attitude **Orbit** Control System (AOCS) in den Unterlagen zum TET-1 bezeichnet. TET-1 nimmt jedoch grundsätzlich keine Orbitwechsel vor.

- Zwei kaltredundante Gyroskope
  (IMU, Englisch: Inertial Measurement Unit)
- Zwei kaltredundante Sternenkompasse mit jeweils zwei heißredundanten Kameraköpfen (CHU, Englisch: Camera Head Unit) und einer Datenverarbeitungseinheit (DPU, Englisch: Data Processing Unit)
  (ACS, Englisch: Autonomous Star Compass)
- Zwei kaltredundante GPS-Empfänger mit jeweils einer Antenne
  (Teil der ONS, Englisch: On-Board Navigation)

Jede Ressource ist Teil einer Redundanzgruppe mit zwei identischen Elementen. Dies erfüllt die Anforderung einer 1-fachen Fehlertoleranz in allen Subsystemen.

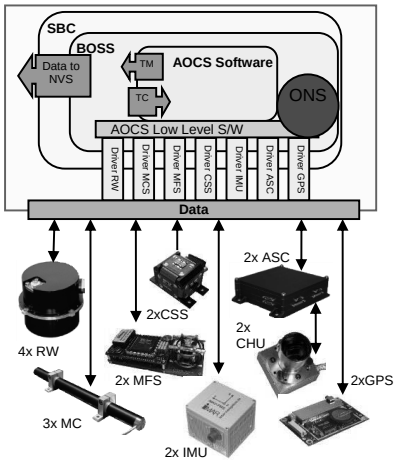


Abbildung 6.2.: Software und Hardware des Lageregelungssystems (Quelle: DLR)

Der Hauptanteil der Erzeugung des Drehmoments wird durch das Reaktionsradsystem aufgebracht. Eine Besonderheit liegt dabei in der Tetraederanordnung der Räder. Dadurch sind bei einem Ausfall eines beliebigen Rades weiterhin alle Achsen im Koordinationssystem abgedeckt. Zusätzlich wird

bei vier verfügbaren Rädern das geforderte Drehmoment auch mit einer geringeren Rotationsgeschwindigkeit pro Rad erreicht. Das Magnetspulensystem wird hauptsächlich zur Endsättigung des Drehimpulses der Reaktionsräder genutzt, kann im Notbetrieb des ACS aber auch alleinstehend geringe Drehmomente erzeugen.

Die Bestimmung von Ausrichtung und Position wird durch die Sensorik ermöglicht. Der TET-1 bietet dabei einer hohe Überlappung von Messdaten zwischen den einzelnen Subsystemen. Die zwei Magnetfeldsensoren bestimmen durch Messung des charakteristischen Erdmagnetfelds die aktuelle Position. Die zwei Sonnensensoren bestimmen die Satellitenausrichtung in Relation zur Sonnenposition. Die Bestimmung des Ausrichtungswinkels zur Erde wird durch die beiden Gyroskope realisiert. Durch einen Vergleich von Sternenkonstellationen zu bekannten Sternenkarten, bestimmen die Sternenkompasse die Orientierung des Systems. Die beiden GPS-Einheiten liefern Informationen zur Position und zum Orbit des TET-1. Zur Übersicht zeigt die Abbildung 6.3 das Hardware-Design zur Platzierung der Ressourcen am TET-1.

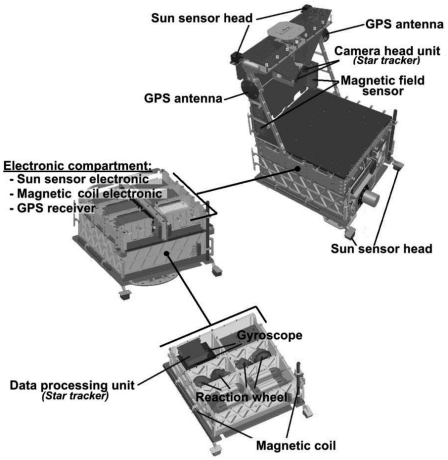


Abbildung 6.3.: Hardware-Design des TET-1 (Quelle: Astro- und Feinwerktechnik Adlershof GmbH)

Die Ressourcen dienen als Senken (Aktuatoren) oder Quelle (Sensoren) für Daten und sind über spezifische Treiber an die Betriebssoftware des ACS angebunden. Die Implementierung des ACS wird auf dem Echtzeit-Betriebssystem BOSS[7] ausgeführt. Das Navigationssystem (ONS, Englisch: On-board Navigation System) ist in das Betriebssystem ausgelagert, wird aber nachfolgend als integraler Teil des ACS angesehen. Die Kommunikation mit dem Bodensegment ist außerhalb des ACS implementiert und über definierte Schnittstellen für Ansteuerung/Telekommandierung (TC, Englisch: Telecommand) und Datenübermittlung/Telemetrie (TM, Englisch: Telemetry) angebunden. Das Betriebssystem und das ACS wird auf vierfach redundanten Hauptrechnern (SBC, Englisch: Satellite Bus Computer) ausgeführt, welche an das Nutzlastversorgungssystem (NVS) angebunden sind.

6.4.4. Architektur der Betriebssoftware

Die Implementierung des ACS basiert auf einer statischen Softwarearchitektur, die in drei Ebenen modularisiert ist. Ein dem Original (unter anderem [YTRM10]) nachempfundene komponentenorientierte Darstellung der Architektur ist in Abbildung 6.4 aufgeführt.

Die Ressourcentreiber werden als Komponenten in der untersten Schicht repräsentiert. Sensoren stellen Daten bereit, Aktuatoren konsumieren Daten. In der Mittelschicht der Architektur befindet sich die EPC-Komponente. Diese realisiert einen Regelkreis zwischen Lagebestimmung, Voraussage von Änderungen und Umsetzung von Änderungsbefehlen. Die Komponente integriert dabei die Daten aus den Sensormessungen (Estimator) und modellbasierte Prognosen zur Lageänderung (Predictor) und stellt Steuergrößen (Controller) für die Aktuatoren zur Verfügung. Zusätzlich ist eine Komponente zum Wechsel zwischen Lageregelungsmodi (ModeProcessor) vorgesehen. Auf oberer Ebene koordiniert die zentrale Komponente ACS die Verarbeitung von Telekommandierungen (Command Interpreter). Weiterhin werden hier Diagnosedaten der Ressourcen, sogenannte Housekeeping-Daten (House Keeper) und deren Gesundheitszustände (Surveillance Monitor) zur Telemetrie-Übertragung bereitgestellt. Für Erprobungen existiert zusätzlich eine gesonderte Komponente zur Testausführung (Testing).

[7] BIRD Operating System Simple; übernommen aus dem BIRD-Satelliten.

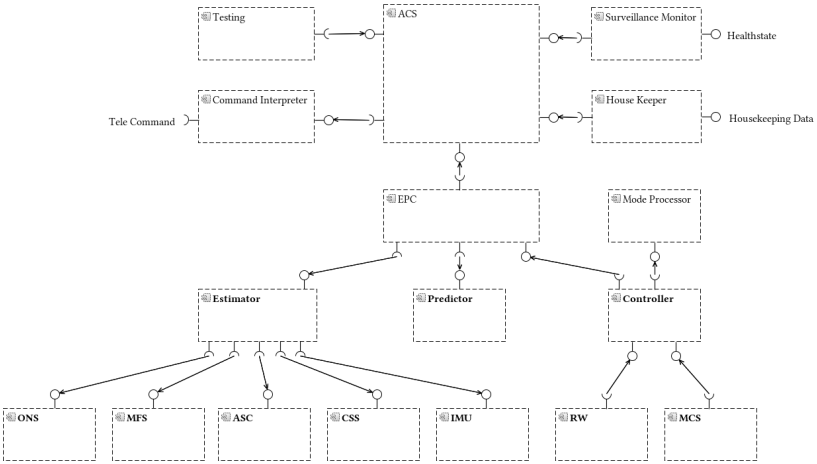


Abbildung 6.4.: Softwarearchitektur des TET-1 ACS

6.4.5. Fehlertoleranzkonzept

Die Bestimmung der notwendigen Redundanz im System richtet sich nach den Kundenanforderungen an die Zuverlässigkeit und den Analysen der Ausfallraten und Zusammenhänge in Fehlerbaum-Analysen. Zusätzlich zur Fehlertoleranz in allen Bauteilgruppen enthält das ACS eine mehrstufige Überwachungslogik als FDIR-Realisierung, vergleiche Abschnitt 2.1.8. Die Umsetzung erkennt und lokalisiert Fehler autonom. Dies beinhaltet auch, die Eingrenzung von Fehlerauswirkungen und dass irrtümlich gemeldete Fehler entsprechend erkannt werden, vergleiche [YTRM10]. Neben kontinuierlichen Überprüfungen auf niedriger Hardwareebene (zum Beispiel Prüfsummen, zeitliche Abweichungen), finden auch auf höheren Ebenen der EPC-Schicht Überprüfungen statt. Dabei werden gemessene und vorausgesagte Werte verglichen, Abweichungen in Toleranzbereichen bewertet und Schnittmengen in Daten unterschiedlicher Sensorik überprüft. Die Fehlererkennung findet in jedem Durchlauf des Regelkreises statt (alle 500ms). Das Konfigurationsmanagement der ACS-Software übernimmt die Isolation

von Fehlern und die Wiederherstellung eines ausführbaren Systemzustands. Dabei können folgende Parameter angepasst werden:

- Satellitenkonfiguration: Spannungsversorgung, Erdschatten, Nutzlast-Aktivität, …
- Aktuator- und Sensorkonfiguration: Power-Flag (on/off), Use-Flag (use/not in use), Health-Flag (healthy/unhealthy)
- Konfiguration der Module Estimator, Predictor und Controller: aktiviert/deaktiviert

Auf Grundlage dieser Wiederherstellungsmaßnahmen wird eine automatische Rekonfiguration des ACS umgesetzt. Dabei werden Health-Flags auf Grundlage der Resultate von Tests auf hoher Ebene autonom gesetzt. Im Anschluss schließt das Lageregelungssystem die ungesunde Ressource aus und kompensiert dessen Ausfall mit Modellwerten. Die Implementierung des ACS nutzt somit die mathematischen Modelle, die als Teil des Predictors implementiert sind. So liegt im System eine funktionale Redundanz[MMT08] zur Behandlung vollständiger Sensorenausfälle vor. Der Wechsel zu einer redundanten Ressource bedarf dem Einschreiten des Wartungspersonals per Telekommandierung. Ein autonomer Wechsel ist ausdrücklich nicht implementiert.

Die Ausführung des gesamten ACS wird ebenfalls überwacht. Der Surveillance Monitor protokolliert relevante Diagnosedaten und überwacht die Prozesse (Englisch: Threads) auf deren zeitliches Verhalten des ACS. Treten Abweichungen außerhalb der Toleranzen auf, wird ein Safe-Modus des gesamten Satelliten ausgelöst und ein Neustart des Hauptrechners angefordert. Im Anschluss werden die Housekeeping-Daten der Regelkreiszyklen während und vor dem Fehlereintritt zur Telemetrie bereitgestellt. Abweichend von einem internen Safe-Modus des ACS, ist der Betrieb des ACS kurzzeitig komplett ausgesetzt.

Die Fehlererkennung und dessen Isolation sind autonom implementiert. Die Behandlung verbleibt bei dem Wartungspersonal. Aufgrund der Risiken einer Systemverlusts bei autonomen Rekonfigurationen, geschieht deren Auslösung grundsätzlich per Telekommandierungen. Primär wird auf physikalische Redundanzen auf Ressourcenebene zurückgegriffen. Sind die homogenen Redundanzen erschöpft, sind umfangreiche Rekonfigurationen unter Nutzung heterogener Gruppierungen vielversprechend. Ermöglicht wird dies

durch teilweise technische Überschnitte in Sensorik (ähnliche Messdaten) und Aktuatorik (ähnliche Auswirkungen).

6.4.6. Bodensegment und Schnittstelle zur Wartung

Wie zuvor am Beispiel des TET-1 eingeführt, ist das Wartungspersonal stark in den Betrieb eines Satelliten eingebunden. Neben dem Satelliten im Weltall (Raumsegment) bilden mehrere Bodenstationen auf der Erde das Bodensegment. Die Abbildung 6.5 zeigt das *FireBIRD-Segment* in vier Bestandteilen.

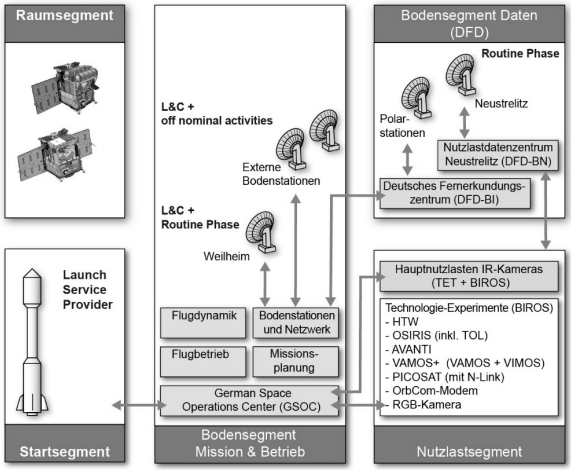


Abbildung 6.5.: FireBIRD-Bodensegment Mission und Betrieb (Quelle: DLR)

Die generelle Steuerung und Überwachung des TET-1 wird vom Kontrollzentrum in Oberpfaffenhofen (GSOC, Englisch: The German Space Operations Center) übernommen. Im Normalbetrieb wird die Bodenstation in Weilheim genutzt. Dadurch sind maximal zwei Kontakte pro Tag zwischen Station und Satelliten möglich. In der Startphase werden weitere externe Bodenstationen hinzugenommen, um eine hohe Frequenz von Kontakten zu erreichen. Dies

trifft auch zu, wenn erweiterte Wartungstätigkeiten nötig sind. Zum Empfang von produktiven Missionsdaten der Nutzlasten und deren Verarbeitung werden die Bodenstation Neustrelitz und weitere polare Stationen genutzt.

Als Schnittstelle zum System stehen dem Wartungspersonal die vorgesehenen Telekommandierungen und die Telemetrie-Daten zur Verfügung. Beide Aspekte sind durch Software-Updates erweiterbar. Die Telekommandierung beschränkt sich im Wesentlichen auf die Anwahl vordefinierter Lageregelungsmodi. Im Wartungsfall können zusätzlich redundante Ressourcen aktiviert werden. Telekommandos werden durch einen Zeitstempel zu einem definierten Zeitpunkt ausgeführt. Telemetriedaten beschreiben den Systemzustand in Form von Housekeeping-Daten. In diesen sind Health- und Use-Flags enthalten. Weiterhin können erweiterte Housekeeping-Daten abgerufen werden, die eine Protokollierung von Systemaktivitäten mit hohem Detailgrad bieten. Der TET-1 kann bis zu 1000 Datensätze (circa eine Woche Betrieb) sammeln, bevor eine Datenübertragung notwendig wird. Dennoch ist eine zeitgesteuerte Telemetrie einmal pro Tag vorgesehen.

Generell liegt ein zeitlich und räumlich begrenzter Zugang zum System vor. Um diese Beschränkungen teilweise zu lockern, wird in der Entwicklung parallel zum produktiven System ein Teststand (Englisch: Engineering Model) aufgebaut. Dieses System bildet die relevanten Elemente des Satellitens im Weltall exakt nach. Tritt im Betrieb die Notwendigkeit eines Software-Updates auf, wird jede Modifikation in mehreren Iterationen auf dem Teststand erprobt. Dies vermindert das Risiko von Folgen fehlerhafter Updates und zudem stehen dem Wartungspersonal zusätzliche Diagnosemöglichkeiten zur Verfügung. So können Auswirkungen vor Ort beobachtet und gemessen werden. Insbesondere werden zusätzliche Messinstrumente an nicht vorhandene Schnittstellen angeschlossen, zum Beispiel Spannungsmessungen innerhalb von Schaltkreisen. Im Betrieb sind die Diagnosemöglichkeiten auf die vordefinierten Messeinrichtungen im System beschränkt. Dies bezieht sich auch auf die Möglichkeiten zum Zugriff auf Speicher, zum Beispiel beim Aktualisieren der Firmware einer Bauteilgruppe.

Zur teilweisen Nachbildung der Umweltumgebungen wurden bei dem Fallstudienpartner eine Plattform für Lageregelungstests [RRG11] aufgebaut. Diese Plattform ermöglicht eine nahezu reibungslose Rotation um 360 Grad und Neigung von bis zu 30 Grad auf einem luftgelagerten Tisch. Durch die Nachbildung des orbitalen Magnetfeldes und des Sonnenstandes sowie einer

freien elektronischen Schwerpunktskalibrierung, sind realitätsnahe Tests möglich. Eine Videodemonstration[8] des Teststands zeigt weitere Details.

6.4.7. Erweiterung und Variation der Lageregelung

Zur Validierung des vorgestellten Ansatzes wird die Architektur des ACS aufgestellt, die neben der physikalischen Redundanz auch softwareseitige Variationen der Redundanzgruppen erlauben. Basierend auf Vorarbeiten [MSH+13, MN14] wurden die Querbeziehungen zwischen Bauteilgruppen gelockert und zusätzliche Variationen in der Sensor- und Aktuatornutzung hinzugefügt, um die Variabilität und Austauschbarkeit im Entwurf zu erhöhen. Die Abbildung 6.6 zeigt den Variationsraum des erweiterten ACS als Feature-Modell [KCH+90], vergleiche Abschnitt 2.2.2. Neben der Lockerung der verbindlichen Existenz diverser Sensorsysteme, wurden für die Subsysteme ASC und CSS jeweils hochauflösende Varianten hinzugefügt. Im ASC ist eine gleichzeitige Aktivierung aller Bauteilgruppen möglich. Das CSS besitzt zwei Abstufungen in der Rechenleistung der Datenverarbeitung. Aufseiten der Aktuatorik ist, abweichend von der gängigen Praxis, der parallele Einsatz beider Magnetspulensysteme erlaubt. Zudem können im Reaktionsradsystem auch minimal gar keine Räder aktiviert sein, was durch eine definierte Nebenbedingung den alleinstehenden Betrieb der Magnetspulen bedingt. Das IMU ist für alle Varianten des Systems zwingend erforderlich.

Statische Modellanalysen in dem Werkzeug FeatureIDE [TKB+14] zeigen, dass aus 30 konkreten und 26 abstrakten Features insgesamt circa 198.000 valide Konfigurationen gebildet werden können. Technisch gesehen basiert das Lageregelungsystems auf 28 physikalischen Ressourcen. Im Rahmen der zusätzlichen Abstufungen in ASC und CSS werden identische Ressourcen unterschiedlich intensiv eingesetzt. Um dies softwareseitig zu adressieren, erhöht sich die Ressourcenmenge virtuell auf 32 Elemente.

Je nach ausgewählter Kombination der Ressourcen erfüllt das System qualitative Anforderungen mit unterschiedlicher Intensität. So ist ein alleiniger Betrieb des Magnetspulensystems zwar energiesparend, jedoch ist die Agilität gegenüber dem Reaktionsradsystem deutlich geringer.

[8] `https://www.youtube.com/watch?v=aZvsXojvZZM`

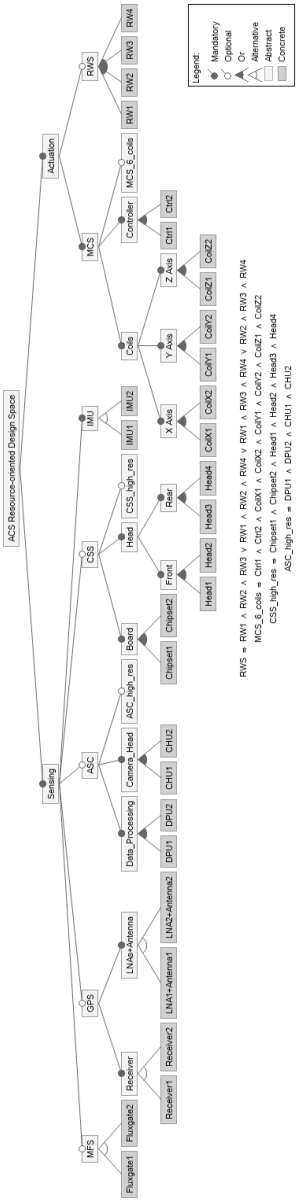


Abbildung 6.6.: Feature-Modell der Ressourcen und Softwarevariationen

Jede Ressource wird in dieser Arbeit durch unterschiedliche Wertebelegungen innerhalb einer Menge von Qualitätsattributen charakterisiert.

6.4.7.1. Qualitätsattribute des ACS

Auf Grundlage diverser Datenblätter[9] der Sensorik und Aktuatorik des ACS, lassen sich insgesamt sechs gemeinsame Qualitätsattribute und deren Wertebelegungen ableiten. Die Sensorik wird auf Grundlage des Signalrauschens (Englisch: *noise*), Messpräzision (Englisch: *accuracy*) und der Lagebestimmungszeit (Englisch: *pointingtime*) beurteilt. Die Aktuatoren haben hauptsächlich einen Einfluss auf die Drehmomenterzeugung (Englisch: *torque*) zur Umsetzung der Agilitätsanforderungen. Beide Ressourcengruppen werden zusätzlich nach deren Energieverbrauch (Englisch: *powerconsumption*) und der erwarteten Betriebsdauer (Englisch: *lifetime*) bewertet. Das Signalrauschen, die Lagebestimmungszeit und der Energieverbrauch sind zu minimieren und die weiteren Attribute zu maximieren. Die Rohdaten der verfügbaren Datenblätter[10] sind für alle Qualitätsattribute in Tabelle 6.1 abgetragen. Neben eindeutigen technischen Spezifikationen für die meisten Attribute, waren einige Werte nur kontextabhängig oder in verschiedenen Werteeinheiten angegeben. Hier wurden Umrechnungen vorgenommen und teilweise Mittelwerte über die möglichen Ausführungskontexte im niedrigen Erdorbit gebildet.

Dies abstrahiert die Qualitätsattribute zwar potenziell stark, jedoch liegt die Authentizität der Werte auch nicht im Fokus dieser Arbeit. Eine hohe Varianz in den Werten, um Ressourcen qualitätsorientiert voneinander abzugrenzen, ist allerdings gegeben.

[9] Beispiel: Datenblatt des Reaktionsrades RW90, `http://www.astrofein.com/2728/dwnld/admin/Datenblatt_RW90.pdf`

[10] Teileweise wurde auf Datenblätter vergleichbarer Bauteile zurückgegriffen, da nicht alle technischen Spezifikationen des ACS öffentlich zugänglich waren.

Tabelle 6.1.: Qualitätsattribute und Wertebelegungen des ACS

Ressource	noise	lifetime	pointingtime	accuracy	torque	powerconsumption
RW		7			0.015	6
MC		10			0.0045	1.11
MFS	30	20	0.2	60		1.5
IMU	20	5	0.3	50		0.8
ASC high	40	30	0.03	70		1.9
ASC low	30	30	0.02	40		1.5
CSS high	50	13	0.04	40		0.7
CSS low	40	13	0.02	30		0.5
GPS	10	15	121.8	90		0.85

6.5. Qualitative Erhebung des Fachwissens von Domänenexperten

Der vorgeschlagene Ansatz ist grundsätzlich domänenunabhängig konzipiert. Eine Validierung sämtlicher Detailaspekte wird jedoch innerhalb einer konkreten Domäne mit technischen Spezifika durchgeführt. Zur Erhebung weitreichender Informationen zu einem praxisrelevanten Entwicklungs- und Wartungsprozess, ist eine Beteiligung von Domänenexperten im Software Engineering unverzichtbar [HA05]. In Abstimmung mit den Experten werden reale Entwurfsentscheidungen und Wartungsaufwände identifiziert sowie weiterhin die Gültigkeit gestellter Annahmen empirisch überprüft.

6.5.1. Systematik des Experteninterviews

Im Rahmen dieser Arbeit werden Experteninterviews durchgeführt. Neben den explorativen Erkenntnisinteressen an den Prozessen, steht die subjektive Beurteilung des Ansatzes sowie die Erhebung zusätzlicher Informationen zu der Fallstudie im Vordergrund der Interviews. Die Interviews werden anhand eines Leitfadens vorbereitet und vorstrukturiert. Der Leitfaden wird dabei nach dem Prinzip der Operationalisierung entwickelt und mit unterschiedlichen Fragetypen abwechselnd strukturiert, vergleiche Abschnitt 2.2.6.

Kommend von den in Abschnitt 1.2.3 aufgestellten Forschungsfragen wird die Operationalisierung der generellen Forschungsfragen hin zu konkreten Interviewfragen durchgeführt. Der Leitfaden, siehe Anhang C, stützt sich auf die nachfolgenden Erkenntniszielsetzungen, Analysedimensionen und Fragekomplexe.

6.5.2. Erkenntniszielsetzungen

Die generelle Zielsetzung der Interviews behandelt nicht die Erhebung quantitativer Daten, sondern den fachlichen Dialog über ausgewählte Aspekte der Anwendungsdomäne. Die Interviews fokussieren dabei insbesondere die folgenden Schwerpunkte.

Nach einem einleitenden Teil zur fachlichen Kenntniserhebung wird eine subjektive Einschätzung der Innovationsbereitschaft der Domäne hinsichtlich einer Intensivierung von autonomen Wartungsaktivitäten vorgenommen. Weiterhin werden Entwurfs- und Wartungsaufwände im existierenden Prozess erhoben. Dabei besteht ein wesentliches Interesse an der Relevanzbeurteilung der Aufwände in Kommunikation und Personal. Beide Aspekte werden weiter mit Bezug auf Datenverfügbarkeit, Latenzen und Prozesstransparenz untersucht. Weiterhin ist eine Einschätzung der Gültigkeit der Annahmen und eingeführter Abstraktionen von Interesse. Abschließend wird die Nutzbarkeit der Ergebnisse im Produktiveinsatz des Ansatzes adressiert.

6.5.3. Festlegung der Analysedimensionen und Ableitung der Fragenkomplexe

In Anlehnung an die Metriken der werkzeuggestützten Messung in Abschnitt 6.6.4, werden die Empirie-bezogenen Validierungsfragen nachfolgend zur Untersuchung systematisch aufbereitet. Dabei wird nach dem Prinzip der Operationalisierung vorgegangen. Zunächst umfasst die konzeptionelle Operationalisierung gemäß Abschnitt 2.2.6 die Definition von Analysedimensionen und Fragekomplexe.

Die Aufstellung der Analysedimensionen umfasst die abstrakte Beschreibung der zu untersuchenden Phänomene. Die Dimensionen besitzen starke Bezüge zu den generellen Forschungsfragen aus Abschnitt 1.2.3 und greifen die empirischen Aspekte dieser auf. Die Fragen werden zunächst in die Analysedimension *Modellannahmen*, *Aufbau Entwurfsraum*, *Quantifizierung Wartungsaufwände*, *Prozessverständnis* und *Akzeptanz des Ansatzes* kategorisiert. Zur weiteren Verfeinerung werden den Analysedimensionen zugleich Fragenkomplexe zugeordnet. Dieser Schritt greift die Beschreibung der Validierungsfragen in Abschnitt 6.3 auf und setzt diese zur qualitativen Beurteilung in den fachlichen Kontext der befragten Experten.

Analysedimension Modellannahmen

Als Prämisse zur Anwendung des Ansatzes stellt Forschungsfrage FF1 die Annahme, dass kontextfreie Fehlerszenarien, festgelegte Betriebsmodi und statische Qualitätseigenschaften gelten und zuverlässige Prognosen über die Systemverfügbarkeit möglich sind. Mit Bezug auf den existierenden Entwicklungs- und Wartungsprozess werden folgende Fragenkomplexe zur Beobachtung dieser Annahmen aufgestellt.

1. Existenz der Kontextabstraktion im Entwicklungsprozess
2. Adäquatheit aktueller Simulationsmodelle
3. Relevanz statischer Datenblattinformationen im Entwicklungsprozess
4. Adäquatheit der Prognose fehlerbehafteter Systemzustände

Die Fragenkomplexe 1 bis 4 behandeln die Validierungsfrage **FF1.1**. Zur qualitativen Bewertung ist die Verwendung und Bedeutung von Prognosemodellen, statischer Daten und Simulationen im aktuellen Entwicklungsprozess zu untersuchen. Zusätzlich wird die Bedeutung von Fehlerannahmen für die Bestimmung des Systemzustands erhoben.

Analysedimension Aufbau Entwurfsraum

Die Anwendbarkeit einer modellbasierten Priorisierung von Rekonfigurationsentscheidungen wird in Forschungsfrage FF2 behandelt. In Bezug auf eine mögliche technische Umsetzung untersucht Forschungsfrage FF4 die Adäquatheit eines vorrangig qualitätsorientierten Vorgehens und dessen Limitierungen. Als Fragekomplexe ergeben sich dabei:

1. Grundlagen und Ausprägungen von Redundanzen
2. Bedeutung eines zentralen modellbasierten Konfigurationsbegriffs
3. Existenz und Verwendung qualitative Degradierungen

Analysedimension Quantifizierung Wartungsaufwände

Diese Analysedimension erkundet die wesentlichen Treiber für Wartungsaufwände nach Forschungsfrage FF2. Insbesondere wird eine Charakterisierung von Personal- und Kommunikationskosten angestrebt. Die Schwerpunkte der Untersuchung sind nachfolgend genannt.

1. Ursprung der wesentlichen Wartungsaufwände
2. Existenz und Umfang von Kommunikationsbeschränkungen
3. Relevanz der Kommunikationsaufwände
4. Ausfallzeiten und Auswirkungen auf Experimente

Analysedimension Prozessverständnis

Die Erhebung des Expertenwissens im Bereich des Entwicklungs- und Wartungsprozesses soll zur teilweisen Beantwortung von Forschungsfrage FF1, Forschungsfrage FF2 und Forschungsfrage FF3 dienen. Dabei liegt das Augenmerk auf den eingesetzten Prognosetechniken zur Fehleranalyse, der Approximation des Systemzustands und der Integration des vorgestellten Ansatzes.

1. Prozessablauf des Software-Updates unter Verwendung von Fehlerprognosen
2. Bedeutung der Transparenz und Nachverfolgbarkeit des Systemzustands
3. Erweiterung existierender modellbasierter Fehleranalysen

Analysedimension Akzeptanz des Ansatzes

Abschließend wird die Akzeptanz des Ansatzes unter den Schwerpunkten Prognose (Forschungsfrage FF1) und modellbasierte Dokumentation (Forschungsfrage FF3) analysiert. Dabei werden die Kontextannahmen und die Intensivierung der Modellbasierung zwecks Anwendbarkeit beurteilt.

1. Akzeptanz eines architekturgetriebenen Vorgehens
2. Gültigkeit von Vorausberechnungen zur Laufzeit
3. Übertragbarkeit in die Praxis

6.5.4. Übergang der Fragenkomplexe zu Interviewfragen

Im Zug der instrumentellen Operationalisierung werden die Fragenkomplexe in Interviewfragen überführt. Die Interviews werden entlang eines Leitfadens, siehe Anhang C, aufgebaut und durchgeführt. Vorgelagert wird die persönliche Situation des Befragten erhoben, abschließend erfolgt eine Rekapitulation der Befragung. Es werden unterschiedliche Fragetypen, vergleiche Abschnitt 2.2.6, eingesetzt, um die Interviews zu strukturieren,

Fakten zu extrahieren, subjektive Einschätzungen einzuholen und Annahmen zu bestätigen beziehungsweise zu widerlegen. Die Fragen teilen sich auf fünf inhaltliche Kategorien auf, welche die Fragenkomplexe der Analysedimensionen aufgreifen.

Fragen zur Konfigurierbarkeit von Systemen: Die Fragen in dieser Kategorie adressieren die Fragenkomplexe aus den beiden Analysedimensionen *Modellannahmen* und *Prozessverständnis*. Neben der Klärung von Begrifflichkeiten zur Systemkonfiguration, werden Annahmen zur Voraussage von Konfigurationsoptionen und umfangreiche Rekonfigurationen im Betrieb des Systems besprochen.

Fragen zur Exploration des Konfigurationsraums: Mit Bezug auf die Festlegung von Konfigurationen sowie homogenen und heterogenen Redundanzgruppen, werden die Fragenkomplexe der Dimension *Aufbau Entwurfsraum* näher untersucht. Im Bereich der Fragenkomplexe zum *Prozessverständniss*, wird der Ablauf der Fehlerbehandlung und die Rolle qualitativer Anforderungen und deren Degradierungen erhoben.

Fragen zum Ablauf des Wartungsprozesses: Erneut werden Fragenkomplexe aus der Dimension *Prozessverständnis* behandelt, allerdings mit Bezug auf den Wartungsprozess. Dabei wird ein Präzedenzfall einer Wartung des TET-1 und der generelle Ablauf der Fehlerbehandlung besprochen. Die Autonomie zur Detektion eines Fehlers, die Rolle von Selbstdiagnosen und die anschließende Ermittlung einer Alternative wird hier beleuchtet. Die Maßnahmen zur Dokumentation von Änderungen an der Konfiguration werden auch thematisiert. Mit Bezug auf die Fragenkomplexe aus *Quantifizierung Wartungsaufwände* werden maßgebliche Limitierungen in der Wartung sowie das Ausmaß von Aufwänden für Personal und Kommunikation erhoben. Im Detail werden die Relevanz von Kommunikationsunterbrechnungen und räumlichen Distanzen als Teil der Befragung eingeordnet. Im dritten Abschnitt wird nach den Komplexen aus der Dimension *Modellannahmen* ermittelt, welche Modelle im Entwicklungs- und Wartungsprozess genutzt und inwieweit diese synchronisiert werden. Dabei liegt ein wesentlicher Fokus auf der Rolle von Architekturentwürfen und der modellbasierten Erprobung von Systemmodifikationen.

Fragen zur Autonomie und Transparenz im Wartungsprozess: Mit einem Schwerpunkt auf Autonomie und Transparenz werden die Fragenkomplexe aus der Dimension *Prozessverständnis* weiter vertieft. Insbesondere werden autonome Behandlungen von kritischen Ausfällen sowie der Einsatz, Varianten und Erweiterungen des ACS Safe-Modus angesprochen. Unter Berücksichtigung der Dimension *Akzeptanz des Ansatzes* und deren Fragenkomplexe wird erfasst, ob teilweise Qualitätsverluste und ein eingeschränkter Betrieb zur Fortsetzung spezifischer Benutzungsmodi akzeptiert werden kann. Weiterhin wird beurteilt, ob eine Entscheidungsunterstützung ohne autonome Ausführung von Modifikationen den Stand der Technik voranbringt. Auf dieser Basis werden Anforderungen zur Begründungen von Rekonfigurationsentscheidungen erhoben.

Fragen zur Anwendbarkeit des Ansatzes: Abschließend werden die Fragenkomplexe aus den Dimensionen *Aufbau Entwurfsraum* und *Akzeptanz des Ansatzes* in den Interviews vertieft. Zunächst wird die Beziehung zwischen Architekturentwurf und produktiver Implementierung im existierenden System geprüft. In diesem Zuge wird eine Festlegung der Architektur als zentrale Komponente im Entwicklungsprozess eingeschätzt und die aktuelle Sichtbarkeit von Systemvarianten in der Architektur erhoben. Weiterführend wird die Verwendung von Architekturinformationen zur Fehlerbehebung und Dokumentation von Wartungsentscheidungen besprochen. Die frühzeitige Abwägung von unterschiedlichen Konfigurationsalternativen auf Basis von Fehlerprognosen und resultierende Mehraufwände im Entwurf ist Teil dieser Fragenkategorie. Dabei findet zugleich eine Beurteilung der Verfügbarkeit von notwendigen Informationen vor der Betriebsphase und die resultierende Unschärfe von Prognosen statt. Abschließend wird diskutiert, ob eine architekturbasierte Analyse und Dokumentation in vorhandene Prozesse integrierbar ist und welche Vor- und Nachteile dabei resultieren.

6.5.5. Auswahl der Domänenexperten

Die Expertenauswahl wurden nach den drei Kriterien von Gläser und Laudel [GL10] durchgeführt, vergleiche Abschnitt 2.2.6. Zunächst ist zu klären, welcher Experte den notwendigen (1) Zugriff auf relevante Informationen

hat, wie hoch die (2) Qualifizierung des Befragten ist und ob dieser (3) bereit ist sein Wissen bereitzustellen.

Zur Untersuchung der Fallstudie TET-1 ACS kommen Personen mit engen Bezügen zu dem On-Orbit-Verifikation-Projekt in die nähere Auswahl. Wesentliche Beteiligte waren das DLR, die Kayser-Threde GmbH und die Astro- und Feinwerktechnik Adlershof GmbH (kurz: Astro), vergleiche Abschnitt 6.4.1. Letzteres Unternehmen hat diverse Komponenten des ACS geliefert, die Systemintegration und Systemtests übernommen. Weiterhin ist das Unternehmen für die laufende Softwarewartung zuständig. Dies garantiert einen weitreichenden Zugriff auf relevante Informationen.

Die Arbeitsgruppe um den TET-1 ist relativ klein und besitzt folglich einen hohen Qualifizierungsgrad. Insbesondere im Bezug auf Lageregelungssysteme kann Astro auf ein umfangreiches Portfolio und Expertise zurückgreifen. Folglich sind Mitarbeitende aus diesem Unternehmen für die Befragung geeignet.

Der TET-1 wurde staatlich gefördert und von den Beteiligten wissenschaftlich in Veröffentlichungen vorgestellt. Insbesondere die Mitarbeitenden von Astro stehen im wissenschaftlichen internationalen Austausch. Entsprechend liegt eine hohe Bereitschaft in der Weitergabe von freigegebenen Informationen für Forschungsprojekte.

Es standen bei dem industriellen Partner ein(e) Systemingenieur(in) und ein(e) Softwareingenieur(in) zur Verfügung um die Interviews durchzuführen. Beide Experten hatten tiefgehende Kenntnisse in den Bereichen Fallstudienwissen und domänenspezifische Prozesse in Entwicklung und Wartung von fehlertoleranten Raumflugkörpern. Insbesondere waren sie mit der Funktions-, Software- und Hardwareentwicklung sowie dem Betrieb des Lageregelungssystems des TET-1 vertraut.

6.5.6. Rahmenbedingungen der Durchführung

Die Interviews wurden in zwei Durchgängen im Februar 2017 in zwei ausführlichen Einzelgesprächen mit visuellem Begleitmaterial durchgeführt. Das erste Interview wurde am 10. Februar 2017 persönlich in den Räumen des industriellen Partners durchgeführt. Das zweite Interview wurde am 21. Februar 2017 in Form einer Videokonferenz abgehalten. Im Vorgespräch

wurde die Verwendung der Ergebnisse und die Möglichkeit von Zensuren und punktuellen Anpassungen der Interviews schriftlich fixiert. Nach einem einleitenden Vortrag zur Vorstellung des entwickelten Ansatzes wurde das jeweilige Interview gemäß Leitfaden durchgeführt. Der Umfang der Interviews bestand aus 40 beziehungsweise 53 Minuten Einleitung, 102 beziehungsweise 111 Minuten Hauptteil und 3 beziehungsweise 16 Minuten Abschluss. Die Abschlussphase des zweiten Interviews enthielt eine Rekapitulierung und einen Abgleich der Erkenntnisse des ersten Gesprächs und war somit ausführlicher.

6.5.7. Methodisches Verfahren zur Transkribierung der Antworten aus den Interviews

Eine Rückführung der Antworten im Experteninterview zu den Analysedimensionen und Validierungsfragen erfolgt in der nachfolgenden Auswertung. Abweichend von dem vierstufigen Verfahren innerhalb der werkzeuggestützen Metriken, wird hier das Verfahren der qualitativen Inhaltsanalyse angewandt, vergleiche Abschnitt 2.2.6. Dabei wird zunächst ein Transkript der Interviews erstellt und anschließend ein Bezug zwischen den Aussagen und Validierungsfragen erzeugt.

Hier wird dabei ein tabellenbasiertes Verfahren zur Dokumentation der Antworten genutzt. Neben der Frage in Kurzform und der zusammengefassten Antwort wird die Quelle (1. beziehungsweise 2. Interview, laufende Minute), die Fragenkategorie sowie adressierte Fragenkomplexe und Analysedimensionen festgehalten. Das Transkript der Interviews ist in Anhang D aufgeführt.

6.5.8. Qualitative Inhaltsanalyse

Gemäß *GQM4* und *GQM5*, vergleiche Abschnitt 6.1.2, werden im Anschluss die Kernaussagen der Interviews erfasst und über die Analysedimensionen auf die Validierungsfragen abgebildet. Abweichend vom chronologischen Verlauf der Interviews und dem zugehörigen Transkript in Anhang D, wird im Folgenden nach Reihenfolge der Validierungsfragen vorgegangen. Die

Expertenaussagen werden dabei, sofern nicht separat gekennzeichnet, im Konjunktiv zusammengefasst wiedergegeben.

6.5.8.1. Analyse für Validierungsfrage FF1.1

FF1.1 *„Ist eine Abstraktion des Laufzeitkontextes im realen Systementwurf legitim?“*

Die Antworten der 13. und 14. Frage aus der Kategorie *Ablauf des Wartungsprozesses* geben Ausschluss darüber, inwieweit von dem Kontext zur Laufzeit des Systems aktuell abstrahiert wird. Die Fragenkomplexe aus der Analysedimension *Modellannahmen* können dabei wie folgt belegt werden. Aus Tabelle D.22 geht hervor, dass auch eine abstrakte Argumentation für eine Rekonfiguration die Objektivität in der Wartung steigern könne. Durch eine Vielzahl von Experten entstünden auch eine Vielzahl von Lösungsvorschlägen, die durch den Ansatz bestätigt oder widerlegt werden könnten. Weiterhin bestätigen die Aussagen aus Tabelle D.23, dass bereits im aktuellen Prozess von dem Laufzeitkontext teilweise abstrahiert würde und ein Umgebungsmodell, vorrangig zur Steigerung der Präzision in Testaufbauten, genutzt werden könne.

6.5.8.2. Analyse für Validierungsfrage FF1.2

FF1.2 *„Ist der Systemzustand nach Fehlereintritt eindeutig beschreibbar?“*

Bei den Fragen bezüglich der Analysedimension *Prozessverständnis* wurden Informationen zum Ablauf des Wartungsprozesses eingeholt. Daraus wird ermittelt, inwieweit der Systemzustand im Bodensegment nachgebildet werden kann. Aus den Antworten der Frage drei in der Kategorie *Exploration des Konfigurationsraums* geht hervor, dass die Reaktion des Systems durch Strategien zur Fehlerbehandlung umfangreich beschrieben sei, vergleiche Tabelle D.6. Somit könne zumindest für jeden erwarteten Fehlerfall die Auswirkung auf das System prognostiziert werden, sobald Erwartungswerte von Messwerten abweichen. Die Daten aus Tabelle D.12 zur Frage 3 aus *Ablauf des Wartungsprozesses* beschreiben weiterhin, dass diese Strategien maßgeblich von dem Wissen und der Erfahrung der Experten abhängen. Die Zusammenhänge zwischen Ursache und Auswirkung auf den Systemzustand würden durch Erprobungen auf dem Teststand manuell erhoben.

6.5.8.3. Analyse für Validierungsfrage FF1.3

FF1.3 *„In welchem Umfang werden Fehlerauswirkungsanalysen durchgeführt?“*

Bei Steigerung des Verständnisses in den Dimensionen *Aufbau Entwurfsraum* und der Überprüfung von *Modellannahmen*, stehen zwei Aspekte im Entwicklungs- und Wartungsprozess im Vordergrund.

Als Teil der Kategorie *Exploration des Konfigurationsraums* vertieft die 2. Frage wie Redundanzen in der Fehleranalyse festgelegt werden. Die Antworten verweisen auf die Fehlerbaumanalyse als gängige Methodik, vergleiche Tabelle D.5. Die Anforderungen des TET-1 fordern einen bestimmen Grad an Fehlertoleranz, dieser müsse in allen Teilen der Modelle gewährleistet werden. So leisten die Reaktionsräder im Idealzustand deutlich mehr Drehmoment als erforderlich, jedoch wäre auch ein nahezu uneingeschränkter Betrieb nach einem Ressourcenausfall möglich. Für den Hauptrechner läge sogar eine vierfache Fehlertoleranz vor. Die Auswahl von Redundanzen wird, laut Aussage der Experten, durch die Flugerfahrung massiv beeinflusst. Neben verrechneten Informationen aus den Datenblättern, wären Kennwerte und Ausfallraten im Systemverbund nachzuweisen.

Andererseits thematisiert die 11. Frage zum *Ablauf des Wartungsprozesses*, vergleiche Tabelle D.20, die Rolle existierender Vorhersagemodelle. Dabei stellt sich heraus, dass ein deutlicher Fokus auf dem hardwarenahen Testen liegt und Softwaremodelle nur dort zum Einsatz käme, wo Situationen physikalisch nicht nachzubilden sein. Hardwaremodelle existieren als Unterstützung von Bauteilen beim Kunden, ansonsten lägen noch Modelle zur Nachbildung von Umweltbedingungen (Atmosphärenreibung, Gravitation, …) vor. In der Praxis werden Hardwareausfälle vorrangig auf der Testplattform wiederholt physikalisch simuliert, teilen die Experten mit.

6.5.8.4. Analyse für Validierungsfrage FF1.4

FF1.4 *„Ist eine statische Betrachtung von Qualitätseigenschaften ausreichend?“*

Diese Auswertung adressiert die ausschließlich statische Betrachtung von Qualitätseigenschaften innerhalb der Analysedimension *Akzeptanz des Ansatzes*. Die Fragen 9 und 10 aus der Kategorie *Anwendbarkeit des Ansatzes*

der Interviews handeln von der Verfügbarkeit qualitativer Prognosen und der Akzeptanz von potenziell unscharfen Meta-Heuristiken.

Die Verfügbarkeit sei nach Tabelle D.44 vorrangig abhängig von technischen Spezifikationen der Ressourcen, da Folgerungen zumeist daraus mathematisch aggregiert werden könnten. Insbesondere seien Datenblätter flugerprobter Ressourcen sehr detailliert. Missionsanforderungen ließen sich ebenfalls auf konkrete Kennwerte reduzieren. Ein Abgleich sei somit rein modellbasiert in frühen Entwurfsphasen grundsätzlich möglich. Laut der Expertenaussagen, verfeinern mit fortschreitender Entwicklung zusätzliche Simulationen von Störungen die Voraussagequalität, um Schwellwerte für Unschärfen zu verringern.

Bezüglich der Akzeptanz von Meta-Heuristiken zeigen die Ergebnisse in Tabelle D.45, dass auch unscharfe Lösungen den Wartungsprozess unterstützen können. Insbesondere in der Ideenfindung sei ein explorativer Ansatz sehr nützlich. Bedingung dafür wäre, dass eine Nähe zu üblichen Lösungen sichtbar sei und Vorschläge nachvollziehbar begründet würden.

6.5.8.5. Analyse für Validierungsfrage FF2.1

FF2.1 *„Hängen personelle Aufwände primär von Diagnose- oder Reparaturtätigkeiten ab?“*

Die Ursachen personeller Aufwände in der Dimension *Quantifizierung Wartungsaufwände* werden in dieser Auswertung abgegrenzt. Dabei werden die 6. und 7. Frage aus der Kategorie *Ablauf des Wartungsprozesses* ausgewertet.

Zunächst betrachten wir entlang der Daten aus Tabelle D.16 die charakteristischen Gründe zur Limitierung der Fernwartung. Als generelle Herausforderungen werden von den Experten die Vorauswahl an Messpunkten in den Housekeeping-Daten, der fehlende Hardwarezugriff für unvorhergesehene Messungen, die Verfügbarkeit von Wartungspersonal und wenige tägliche Kontakte zum System herausgestellt.

Zur Präzisierung des Personalaufwands geben die Antworten aus Tabelle D.17 Aufschluss darüber, dass es keinen Bereitschaftsdienst gäbe und dass bei einem Fehlerfall die Anzahl beteiligter Personen schnell ansteigen könne, wenn mehrere Subsysteme betroffen seien. Durch sehr aufwendige Prozesse

und Aufgabenverteilungen sei jedes Software-Update dadurch sehr zeit- und personalintensiv. Neben der Personalverfügbarkeit müsse auch das jeweilige Wissen vorhanden sein. Ein Experte erwähnt, dass beim TET-1 die Reparatur eines Problems beispielsweise fast ein halbes Jahr angedauert hat.

6.5.8.6. Analyse für Validierungsfrage FF2.2

FF2.2 *„Resultiert der wesentliche Kommunikationsaufwand aus Datenmengen oder Latenzen im Systemzugriff?"*

Zur Beantwortung der wesentlichen Ursachen für Kommunikationsaufwände wird in der Dimension *Quantifizierung Wartungsaufwände* die Frage neun aus der Kategorie *Ablauf des Wartungsprozesses* herangezogen.

Durch die unterbrechungsbehaftete Kommunikation zum Satelliten entstünden laut Tabelle D.18 massive Beeinträchtigungen. Durch die maximal zwei möglichen Kontakte im Regelbetrieb des TET-1 könnten Fehler so im schlimmsten Fall erst 12 Stunden nach dem Auftreten beobachtet werden. Die anschließende Phase zur Fehlerbehebung sei entsprechend kurz oder verlängert sich schnell um mindestens einen halben Tag. Das System müsse daher eine Rückfallebene bieten; im TET-1 als Safe-Modus realisiert. In diesem Fall sei ein produktiver Betrieb allerdings nicht mehr möglich. Die Kapazitäten zum Upload sind zwar beschränkt, da Updates aber auf mehrere Orbits verteilt werden, sei dies unkritisch. Die wesentlichen Beschränkungen in der Kommunikation lägen also in der Latenz und würden zugleich eine möglichst schnelle Identifikation von Fehlerbehandlungen fordern. Die Hinzunahme weiterer kommerzieller Bodenstationen würden weitere Kontakte ermöglichen, seien jedoch mit Kosten für jeden Kontakt verbunden.

6.5.8.7. Analyse für Validierungsfrage FF2.3

FF2.3 *„Lässt sich anhand vorliegender Diagnosedaten die verbleibende Konfigurierbarkeit des Systems nachvollziehen?"*

Mit starkem Fokus auf die Dimension *Prozessverständnis* und zusätzlicher Betrachtung der *Quantifizierung Wartungsaufwände* werden vier Fragen hier ausgewertet.

In der Kategorie *Ablauf des Wartungsprozesses* ermittelt Frage 4 (Tabelle D.13), dass Lösungen für Konfigurationswechsel grundsätzlich auf aufwendigen Simulationen von Szenarien auf der Testplattform basieren würden. Eine vorsorgliche Untersuchung fände nicht statt. Weiterhin erhebt die 10. Frage aus dem gleichen Abschnitt die Limitierungen beim Systemzugriff, vergleiche Tabelle D.19. Dabei wird vor allem die Tatsache der beschränkten Schnittstellen zur Auswertung physikalischer Größen und Durchführung von Software-Updates von den Experten genannt. Ein direkter Zugriff auf die Bauteile sei mangels eigener Telekommunikationseinheiten nicht möglich. Beim TET-1 wäre für ein Update eine softwareseitige Anpassung des Hauptrechners notwendig, um die Firmware der Reaktionsräder per Telekommandierung zu aktualisieren. Weiterhin sei es nicht möglich ständig alle Subsysteme auf Verfügbarkeit zu überprüfen und so Prognosen über den verbleibenden Konfigurationsraum zu treffen.

Die Fragen 10 und 11 der Kategorie *Autonomie und Transparenz im Wartungsprozess* liefern weitere Details zum Verständnis autonomer Wechsel von Lageregelungsmodi. So ergibt sich aus Tabelle D.33, dass im TET-1 der Surveillance-Monitor für das selbstständige Kommandieren des Safe-Modus zuständig sei. Weitere autonome Wechsel seien in der Regel nicht vorgesehen. Eine Ausnahme würde ein Wechsel in den Modus zur groben Sonnenausrichtung bilden, sobald das System die inertiale Orientierung verlieren würde, vergleiche Tabelle D.34.

6.5.8.8. Analyse für Validierungsfrage FF2.4

FF2.4 *„Ist ein modellbasierter Konfigurationsbegriff aus der Entwurfsphase für die Wartungsanalyse zur Laufzeit einsetzbar?“*

Mit einem Fokus auf einen modellbasierten Konfigurationsbegriff, werden nachfolgend die Dimensionen *Aufbau Entwurfsraum* und *Akzeptanz des Ansatzes* untersucht.

Zunächst wird in der ersten Fragestellung zum Thema *Exploration des Konfigurationsraums* die Festlegung von Konfigurationen im Entwicklungsprozess beleuchtet. In Tabelle D.4 wird entsprechend argumentiert, dass sich Auswahl und Kombination von Ressourcen nach dem Missionsziel und den Zuverlässigkeitsanforderungen richten würden. Bei TET-1 bilden sich, laut

der Experten, die Konfigurationen aus der Summe der elf Experimente und den Erfahrungen aus dem Vorgängersystem. Weiterhin sei ein Bauteileausfall in allen Subsystemen erlaubt.

Ebenfalls als erster Punkt in der Kategorie *Anwendbarkeit des Ansatzes* werden die Beziehungen zwischen Architekturentwurf und produktiver Implementierung in den Interviews thematisiert. In der Auswertung von Tabelle D.36 ergibt sich, dass Architekturen zwar im Entwurf vorliegen, allerdings im Betrieb nicht aufgegriffen würden. Änderungen seien in der Architektur potenziell nicht sichtbar und würden auch nicht nachgetragen. Wesentliche Teile von Architekturen wären jedoch aus der Strukturierung des Quelltextes ableitbar. Dies fände üblicherweise aber nicht statt.

Die elfte Frage aus der vorherigen Kategorie untersucht die Integration einer Dokumentation in den Wartungsprozess auf Architekturbasis. Die Antwort, vergleiche Tabelle D.46, macht deutlich, dass die architekturbasierten Konfigurationsvorschläge den Entwickler produktiv unterstützen würden, sofern dieser mit dem konzeptionellen Software Engineering vertraut sei.

6.5.8.9. Analyse für Validierungsfrage FF3.3

FF3.3 *„Ist der vorgestellte Ansatz in bestehende Wartungsprozesse zur Fehlerbehandlung integrierbar?“*

Zur Beurteilung der Integration des Ansatzes, wird der existierende Wartungsprozess und der Angemessenheit des Ansatzes in insgesamt sechs Fragen, in den Dimensionen *Prozessverständnis* und *Akzeptanz des Ansatzes*, behandelt.

Die Antworten auf die zweite Frage zum *Ablauf des Wartungsprozesses* aus Tabelle D.11 erläutern den idealtypischen Ablauf des Wartungsprozesses. Dabei wird deutlich, dass das System grundsätzlich in der Lage sei autonom Zeiträume bis zur nächsten Kontaktaufnahme oder bis zum Ende einer Wartungsperiode zu überbrücken. Weiterhin wird erläutert, dass zwar wohl-getestete Anpassungen an Algorithmen oder Strategien vorgenommen würden, neue Methoden zur Fehlerbehandlung oder Lageregelungsmodi aber nicht hinzugefügt würden.

Zur Einschätzung der Akzeptanz eines autonom agierenden Ansatzes werden zwei Varianten in den Fragen fünf und sechs, unter dem Gesichtspunkt

Autonomie und Transparenz im Wartungsprozes, beurteilt. Die erste Variante mit gleichbleibender Systemqualität, vergleiche Tabelle D.28, wird als positiv von den Experten aufgenommen. Eine Abwandlung mit sinkender Qualität ohne Totalausfall, vergleiche Tabelle D.29, wäre allerdings nur als sinnvoll erachtet, wenn ein Mehrgewinn gegenüber dem Safe-Modus sichtbar sei. Die resultierenden Messdaten müssten dabei für den Anwender gesondert ausgezeichnet werden. Die anschließende 7. Frage, aus der gleichen Kategorie, untersucht die Akzeptanz des Ansatzes zur Entscheidungsunterstützung, ohne autonome Ausführung von Aktionen. Nach Tabelle D.30 sei diese Ausprägung des Ansatzes vor allem für die Entscheidungsträger und Kunden interessant, da somit neben einer Übersicht über das System auch Begründungen für Änderungen verdeutlicht würde.

Die Anwendung des Ansatzes steigert die Kosten zur Entwurfszeit. Dieser Aspekt wird in der Frage acht aus Kategorie *Anwendbarkeit des Ansatzes* betrachtet. Die Experten teilen mit, dass eine Akzeptanz vor allem mit dem Grad der Interaktion mit dem Anwender steht oder fällt, vergleiche Tabelle D.43. Einerseits sei eine schnelle Reaktion des Systems gewünscht, andererseits sei die Domäne konservativ gegenüber autonomer Funktionen eingestellt. Eine Integration als Teil der ausführlichen Fehleranalysen zur Steigerung der Ausfallsicherheit erscheint den Experten als möglich. Für kommerzielle Missionen sei die Integration hingegen erschwert, da dort eine möglichst kurze Konzeptphase den Systementwurf bestimmt.

Abschließend werden die Vor- und Nachteile in der zwölften Frage zur Akzeptanz geklärt. Hierbei treten unterschiedliche Einschätzungen in Tabelle D.47 auf. Zunächst wird die praktische Verwendung des Ansatzes positiv bewertet. Bei der Exploration vieler Lösungsalternativen könne der Ansatz helfen. Allerdings würden im aktuellen Prozess auch bereits eine Vielzahl von Fehlerkombinationen und deren Auswirkungen observiert. Dennoch bliebe die Abwägung der Alternativen zueinander sehr aufwendig und einige Varianten so vermutlich unentdeckt. Insbesondere für frühe Designstudien wäre der Ansatz gut einsetzbar. Zum späteren Betrieb seien aber diverse Erprobungen der ermittelten Konfigurationen immer verpflichtend notwendig. Ein weiterer Vorteil des Ansatzes wurde im Manifestieren des Expertenwissens in einem Modell gesehen. Insbesondere durch lange Laufzeiten der Systeme seien ehemalige Entwickler häufig nicht mehr erreichbar. Da sei eine digitale Wissensbasis sehr nützlich. Dabei wäre es zwingend notwendig das Modell auch zur Laufzeit mit weiteren Daten zu versorgen, um Voraussagen

zu verfeinern. Die Nutzung von Datenblättern sei weiterhin nicht abwegig, könne aber durch zusätzliche Daten aus dem Laufzeitkontext optimiert werden. Eine Erweiterung zur Ausführung autonomer Rekonfigurationen, wäre sehr wertvoll für die Domäne. In jedem Fall müsse das Projektmanagement dieser Innovation positiv gegenüberstehen und notwendige Zusatzinvestitionen genehmigen. Dies würde in der hochspezialisierten Domäne präzisen Begutachtungen erfordern.

6.5.8.10. Analyse für Validierungsfrage FF3.4

FF3.4 *„Ist eine vorrangig modellbasierte Dokumentation der Wartungsschritte zielführend?“*

Zur Beurteilung der Angemessenheit einer modellbasierten Dokumentation in der Wartung, werden die Antworten aus den Kategorien *Ablauf des Wartungsprozesses* und *Autonomie und Transparenz im Wartungsprozess* zusammengezogen. Für die Analysedimension *Prozessverständnis* liefert die Tabelle D.14 Hinweise darauf, dass aktuell eine Protokollierung der Wartungsaktivitäten nur auf textueller Ebene auf Basis der Housekeeping-Daten stattfindet. Eine Festlegung auf eine Notation sei nur bedingt möglich, da die Fehler sehr vielfältig seien. Weiterhin fände die Wartung verteilt statt, wodurch ein Austausch zwischen den Entwicklern notwendig sei.

Bezogen auf die Dimension *Modellannahmen* zeigt die 12. Frage zum Ablauf des Wartungprozesses, dass Änderungen in der Systemkonfiguration nicht auf Architekturebene sichtbar seien, vergleiche Tabelle D.21.

Eine Steigerung der Dokumentation des Rekonfigurationsprozesses wird gemäß Tabelle D.31 als sinnvoll eingestuft, was die Dimension *Akzeptanz des Ansatzes* unterstützt. Dabei wird auf die achte Frage in der *Autonomie und Transparenz im Wartungsprozess* geantwortet, dass sich ein explorativer Ansatz erst ab einer Vielzahl von Konfigurationen lohnen würde.

6.5.8.11. Analyse für Validierungsfrage FF4.2

FF4.2 *„Lassen sich Differenzen von Konfigurationen auf rein struktureller und qualitativer Ebene adäquat beurteilen?“*

Im Folgenden wird das *Prozessverständnis* in der Fragekategorie *Exploration des Konfigurationsraums* adressiert. Die Fragen 5 und 6 untersuchen die Existenz, beziehungsweise die Rolle qualitativer Degradierungen und Anforderungen im existierenden Entwicklungsprozess. Aus Tabelle D.8 geht hervor, dass qualitative Abstufungen bei der Bestimmung der inertialen Lage vorgesehen seien, diese aber nicht im Normalbetrieb genutzt würden. Zur Überbrückung von Sensorausfällen würden weiterhin über einen begrenzten Zeitraum Modellwerte genutzt, um einen autonomen Betrieb bis zur nächsten Kontakteaufnahme zu gewährleisten. Laut der Experten beschreiben die Missionen qualitative Mindestanforderungen, welche das Design des Satelliten erfüllen und mit Redundanzen absicheren müsse, vergleiche Tabelle D.9. Abweichungen in den Systemqualitäten seien somit möglich und sichtbar, würden aber nur im Notbetrieb genutzt.

Bezüglich der Dimension *Akzeptanz des Ansatzes* weisen die Antworten auf Frage vier aus der Kategorie *Anwendbarkeit des Ansatzes* darauf hin, dass sich die Beurteilung der Systemqualität nach der Verfügbarkeit von Redundanzen richtet würden. Somit würden eine strukturelle Änderung durch Ausfälle, vergleiche Tabelle D.39, auch auf eine Rekonfiguration hinweisen.

6.6. Werkzeuggestützte Modellierung und Erhebung der Analysedaten

Die domänenspezifische Entwurfsraumexploration der Fallstudie, vergleiche Abschnitt 6.4 entlang des vorgestellten Ansatzes, erfordert die Modellierung des Systems sowie die Definition von Betriebsmodi, Analysedimensionen und zugehörigen Metriken. Die Messungen erfolgen auf Basis einer erweiterten Variante des Systems mit einem erhöhten Grad an Redundanz und Variabilität in Sensorik und Aktuatorik.

6.6.1. Modellierungs- und Analyseprozess

Die Modellierung erfolgt in einer Auswahl von Werkzeugen für den variantenreichen Entwurf und die Analyse von komponentenbasierter Architektu-

ren. Alle Werkzeuge bauen auf dem Eclipse Modeling Framework (EMF)[11] auf und sind frei verfügbar.

In Abbildung 6.7 ist der Verbund der Werkzeuge skizziert. Dabei werden die Werkzeuge Palladio Bench[12] und das zugehörige Plugin DSE/PerOpteryx[13] sowie das im Rahmen dieser Arbeit weiterentwickelte Werkzeug ArEva[14] eingesetzt.

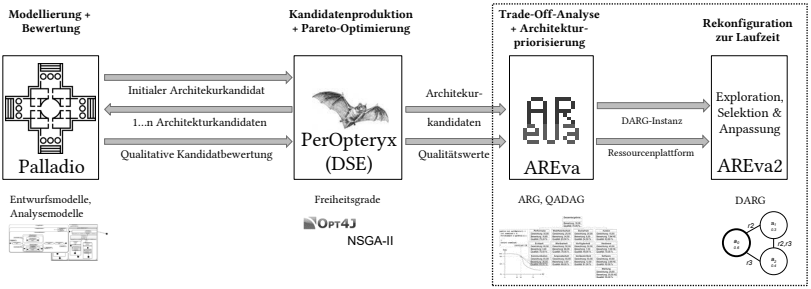


Abbildung 6.7.: Integrierte Werkzeugunterstützung

Zunächst wird die Softwarearchitektur des ACS auf Basis des Palladio Komponentenmodells (PCM, vergleiche Abschnitt 2.2.1), entworfen. Wie in Abschnitt 6.4.4 beschrieben, werden Hardwareressourcen ebenfalls in der Architektur repräsentiert. Im Rahmen einer nicht-invasiven Erweiterung des PCMs, verweisen eine oder mehrere Ressourcengruppen aus einem Modell zur Definition der Ressourcenplattform (*Plattformmodell*), die entsprechende Basiskomponente in der Softwarearchitektur. Jede Ressource wird anhand der Wertbelegungen für die Qualitätsattribute aus Abschnitt 6.4.7.1 spezifiziert. Zudem beschreibt das zusätzliche Modell die Restriktionen zwischen den Ressourcen, vergleiche das Feature-Modell [KCH+90] aus Abschnitt 6.4.7. Auf Basis des PerOpteryx-Ansatzes (beziehungsweise Design Space Exploration, DSE) werden Architekturkandidaten generiert. Der Ansatz stützt sich auf dem Konzept der Freiheitsgrade nach Koziolek [Koz11] zur Festlegung

[11] http://www.eclipse.org/emf
[12] http://www.palladio-simulator.com
[13] https://sdqweb.ipd.kit.edu/wiki/PerOpteryx
[14] http://www.github.com/lmaertin/areva

von Variationspunkten in der initialen Softwarearchitektur. Im vorliegenden Fall werden Freiheitsgrade für die Selektion der Basiskomponenten mit Ressourcenannotationen definiert. Auf dieser Basis instantiiert PerOpteryx eine Kandidatenmenge möglicher Architekturausprägungen. Jeder einzelne Kandidat wird entlang der technischen Spezifikationen seiner gebundenen Ressourcen nach Qualitätskriterien bewertet. Nach erfolgter Einzelbewertung werden die Architekturkandidaten in einem sechsdimensionalen Raum aller Qualitätsattribute angeordnet, der die Zielkonflikte (Englisch: Trade-Offs) zwischen den Attributen beschreibt. Entsprechend einer Pareto-Optimierung, vergleiche Abschnitt 2.2.3, werden Kandidaten identifiziert, die gruppiert eine Pareto-Front bilden. Diese Informationen werden zur Vorauswahl von bestmöglichen Architekturkandidaten genutzt. ArEva erstellt ausgehend von der Kandidatenmenge eine Instanz des *ARG*-Modells, vergleiche Abschnitt 4.1.1. Weiterhin wird das *QADAG*-Modell instantiiert, wobei zunächst eine Gleichgewichtung aller Qualitätsattribute automatisiert gesetzt wird. Die Komponenten der erzeugten Architekturen sind mit Ressourcen verknüpft und tragen qualitative Bewertungen. Eine *Konfiguration* repräsentiert hier die Kombination der Modellartefakte. In den nächsten Schritten des Prozesses definiert der Benutzer hauptsächlich maximale Rekonfigurationskosten und parametrisiert die *QADAG*-Instanz hinsichtlich des anvisierten Betriebsmodus. ArEva erzeugt im Anschluss eine minimierte Instanz des *DARG*-Models entsprechend Abschnitt 5.1. Zur simulierten Laufzeit des Systems verarbeitet der Suchalgorithmus, vergleiche Abschnitt 5.2, die Entscheidungsstruktur und ermittelt nach Vorgabe einer gegebenen Fehlersequenz eine qualitativ optimale Alternativkonfiguration, die mit der noch verfügbaren Hardwarekonstellation umsetzbar ist. Neben der neuen Konfiguration werden diverse Kennzahlen zur Begründung der Auswahl protokolliert.

6.6.2. Erzeugung des Entwurfsraums durch multi-kriterielle Optimierung

Auf Grundlage der PCM-Modellinstanz der Softwarearchitektur des ACS aus Abschnitt 6.4.4 und der Variabilitätsinformationen der Fallstudienerweiterung, vergleiche Abschnitt 6.4.7, werden Freiheitsgrade in der Architektur definiert. Dazu wird zunächst die Variabilität auf Gruppen homogener Redundanzen reduziert. Dies ist möglich, da homogene Beziehungen zwischen

Ressourcen durch Gruppierungen und Mindestmengen im Plattformmodell beschrieben werden. Zur Veranschaulichung zeigt die Abbildung 6.8 zunächst ein Feature-Modell [KCH+90] für den symbolischen Raum der Redundanzgruppen.

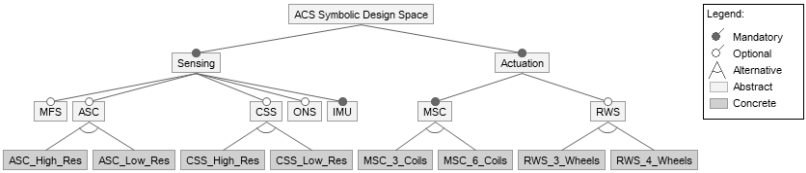


Abbildung 6.8.: Feature-Modell der symbolischen Redundanzgruppen

Die Variabilität wird durch unterschiedliche Design-Optionen innerhalb der Freiheitsgrade zur Komponentenselektion, entsprechend Tabelle 6.2, als Entscheidungsraum festgehalten. Die Optionen verweisen auf alternative Basiskomponenten, die anstelle der Standardkomponente ausgewählt werden können. Die gewählten Optionen sind an Ressourcengruppen im Plattformmodell gekoppelt. Diese wiederum verweisen auf eine Menge von Ressourcen.

Tabelle 6.2.: Freiheitsgrade im erweiterten ACS

Freiheitsgrad	Option 1 (initial)	Option 2	Option 3
MCS	MCS 3 Coils	MCS 6 Coils	-
RW	RW Dummy	RW 3 Wheels	RW 4 Wheels
ASC	ASC Dummy	ASC High Res.	ASC Low Res.
CSS	CSS Dummy	CSS High Res.	CSS Low Res.
MFS	MFS Dummy	MFS	-
ONS	ONS	ONS Dummy	-

Die Option *RW 3 Wheels* verweist auf alle vier Elemente vom Typ *Reaktionsrad* und fordert dabei mindestens drei Elemente. Eine *dummy*-Option verweist auf eine Ressourcengruppe mit keiner Ressource und entspricht

somit der Abwahl sämtlicher Ressourcen der betrachteten Bauteilgruppe. Die vollständige Zuordnung ist in Tabelle A.1 in Anhang A nachzulesen.

Mit PerOpertyx wird eine multi-kriterielle Optimierung[15] im durch die Freiheitsgerade aufgespannten Entscheidungsraum durchgeführt. Dabei verweist der initiale Konfigurationskandidat auf die Standardwerte der Freiheitsgrade. Die Optimierung erfolgt mit einer evolutionären Suche mit 200 Iterationen bei 20 Individuen pro Population und einem Crossover-Faktor von 0,9. Es resultieren 210 gültige Konfigurationen, 16 davon sind Pareto-effizient. Im Weiteren werden die Kriterien in der Optimierung als Qualitätsdimensionen bezeichnet. Zur Veranschaulichung der erzeugten Kandidatenmenge, zeigt Abbildung 6.9 die paarweisen Verhältnisse innerhalb des sechsdimensionalen Raums. Pareto-effiziente Kandidaten sind hervorgehoben. Die Wertestreuung in der Dimension des Drehmoments ist relativ gering, da nur wenige Parameter aus der Aktuatorik diese beeinflussen. Die Abstraktion auf Redundanzgruppen reduziert die Menge der Konfigurationen somit deutlich. Zur Festlegung von einheitlichen Nutzwerten für jede Konfiguration, sind voneinander abweichende qualitative Bewertungen weiterhin gegeneinander abzuwägen.

6.6.2.1. Qualitätsdimensionen und Wertebelegungen

Aus den Qualitätsattributen und deren Belegungen für jede Ressource, manifestiert im Plattformmodell, ergibt eine individuelle Bewertung für jede Konfiguration. Im Weiteren werden diese als Qualitätsdimensionen bezeichnet. Die normalisierten Wertebelegungen aller Kandidaten sind in Tabelle A.2 in Anhang A aufgeführt. Die Abbildung 6.10 zeigt die Qualitäten von vier exemplarischen Konfigurationen aus der Ergebnismenge. Aus den Abweichungen der Werte ist erkennbar, dass je nach Konfigurationsauswahl unterschiedliche Qualitätsdimensionen adressiert werden. Je nach Einsatzzweck sind also variierende Mengen von Konfigurationen optimal.

[15] Eine Nebenstudie dieser Arbeit zeigt, dass durch die Berücksichtigung der Redundanzgruppen für die Fallstudie auch grundsätzlich eine erschöpfende Suche mit akzeptablen Aufwand möglich ist. Aus Gründen der Generalität und zur qualitativen Abgrenzung der Konfigurationsalternativen, wird hier weiterhin eine qualitätsorientierte Optimierung durchgeführt.

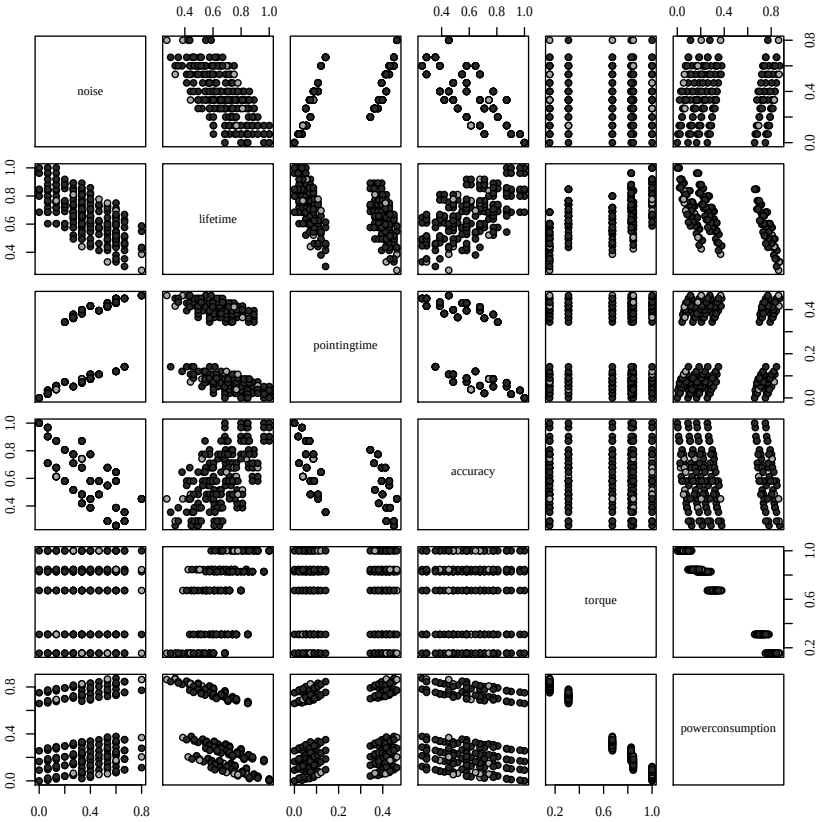


Abbildung 6.9.: Paare über Qualitätsdimensionen nach multi-kriterieller Optimierung

6.6.3. Messaufbau

Die Ergebnisse der multi-kriteriellen Optimierung spannen den vollständigen Rekonfigurationsraums als *ARG* (vergleiche Abschnitt 4.1.1) in ArEva auf. Zur Ableitung betriebsoptimaler Varianten dieses Raums, sind zunächst die *Betriebsmodi* des Systems festzulegen. Weiterhin ist für die Erprobung des Ansatzes eine Fehlersequenz zu definieren.

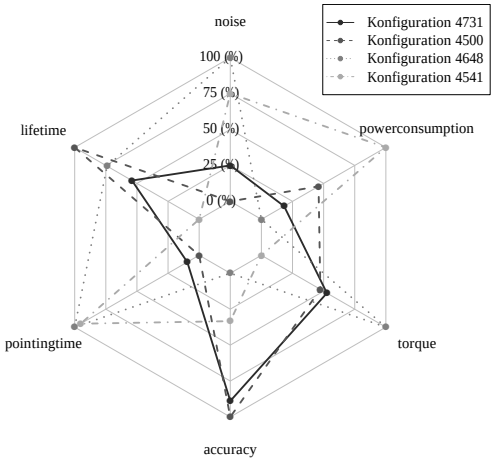


Abbildung 6.10.: Normalisiertes multi-kriterielles Qualitätsspektrum über vier exemplarischen Konfigurationen

6.6.3.1. Experimente als Betriebsmodi und Qualitätsattributsgraphen

Die Betriebsmodi definieren die Verhältnisse der Qualitätsdimensionen zueinander und geben individuelle minimale Akzeptanzwerte vor. Somit ist es möglich jede Konfiguration auf deren Eignung zum Betrieb anhand eines Nutzwerts einzuordnen. Dadurch kann weiterhin eine Auswahl betriebsoptimaler Konfigurationen zur Laufzeit getroffen werden. Im TET-1 werden Betriebsmodi durch Experimente repräsentiert. Entsprechend Abschnitt 6.4.2 definieren die elf Experimente den Missionsplan für das erste Betriebsjahr.

Zur repräsentativen Validierung werden drei Experimente [LHSR12] als Betriebsmodi ausgewählt, die sich in ihren Anforderungen stark unterscheiden. Konkret werden die Lithium-Polymer-Batterie (Experiment $N1$), das Pico-Antriebssystem (Experiment $N7$) und die Infrarot-Kamera (Experiment $N15$) betrachtet. In der Abbildung 6.11 sind die Einbauorte innerhalb der Nutzlast des TET-1 zur Veranschaulichung hinterlegt. Im Folgenden werden die Experimente verkürzt beschrieben und die qualitativen Anforderungen an das ACS hervorgehoben.

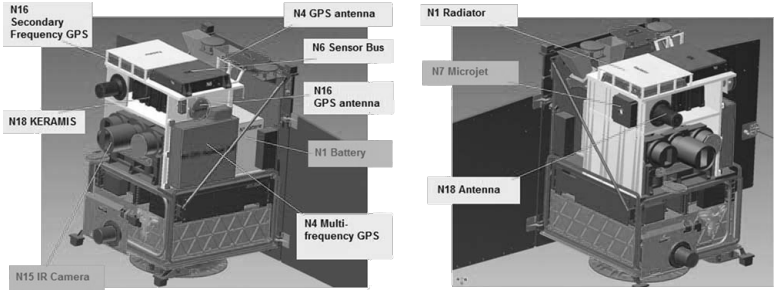


Abbildung 6.11.: TET-1 Nutzlast-Anordnung (Quelle: [LHSR12])

Experiment N1, Lithium-Polymer-Batterie: In diesem Experiment wird das zyklische Laden und Entladen einer neuartigen Lithium-Polymer-Batterie erprobt. Innerhalb der Sonnen-Phase wird das Laden durchgeführt, im Erdschatten die Entladung. Der Vorgang des Ladens weist einen hohen Energiebedarf auf, entsprechend muss das ACS *energieeffizient* sein, vergleiche [AMS$^+$10]. Die Lade-/Entladeströme erzeugen ein elektromagnetisches Feld, welches den Magnetfeldsensor stört. Um eine dennoch präzise Lagebestimmung durchzuführen, ist sonstiges *Signalrauschen* in der Sensorik zu minimieren. Durch die eingegrenzte Bestimmung des Magnetfeldes, kann auch das Magnetspulensystem schlecht austariert werden. Entsprechend sollte ein alleinstehender Betrieb dort nicht stattfinden. Stattdessen ist das Reaktionsradsystem zu verwenden, was qualitativ durch hohe *Agilitätsanforderungen* erzwungen wird. Das Experiment erfordert eine *mittlere Laufzeit*, um adäquate Aufzeichnungen zu Spannung, Strom und Temperatur zu ermöglichen. Die Anforderungen an *Präzision* und *Ausrichtungszeit* sind nebensächlich.

Experiment N7, Pico-Antriebssystem: Ein experimentelles Antriebssystem für Pico-Satelliten wird in diesem Versuch in kurzen Perioden verifiziert. Insbesondere wird die Selbstanpassung zwischen quasi-kontinuierlichem und pulsweisem Betrieb der Schubdüse untersucht. Die Nutzlast erzeugt minimale Drehmomente, deren Ausmaß im Anschluss gemessen wird. Zur Erkennung der minimalen Lageänderungen und zur Vermeidung der Kontamination der Solarpaneele mit dem ausgestoßenem Treibstoff [AMS$^+$10], müssen die *Präzision* und *Lagebestimmungszeit* des ACS hoch sein. Zur Vermeidung

von Messfehlern ist weiterhin ein reduziertes *Signalrauschen* wichtig. Um die Erzeugung der Drehmomente zu beurteilen, sind sonstige Aktuatoren des ACS zu deaktivieren, entsprechend liegen keine *Agilitätsanforderungen* vor. Weiterhin ist das Experiment nur von geringem *zeitlichen Ausmaß* und benötigt wenig *Energie*.

Experiment N15, Infrarot-Kamera: Dieses Experiment dient zur Erprobung einer neuartigen Datenaufbereitung zur Analyse von Hochtemperaturvorfällen auf der Erde. Zur Erdbeobachtung nutzt das System drei Kameramodule. Diese sind identisch zu einem Vorgängersystem, jedoch findet im TET-1 zusätzlich eine Vorauswertung unmittelbar in der Nutzlast statt. Zur Ausweitung der Schwadbreite[16] rotiert TET-1 periodisch. Dies bedingt eine hohe *Agilität* im ACS. Zur Gewährleistung adäquater Fotoaufnahmen, muss eine stabile Lage durch ein ausgewogenes Verhältnis von *Präzision*, *Signalrauschen* und *Lagebestimmungszeit* gelten [AMS+10]. Dabei sollte ebenfalls der Energiebedarf ausgewogen sein. Bezüglich der späteren produktiven Verwendung der Nutzlast als Teil der FireBIRD-Mission, sollte die *Lebenszeit* grundsätzlich hoch sein.

Aus den genannten Betriebsanforderungen an das ACS werden pro Experiment Gewichtungen und minimale Akzeptanzwerte für jede Qualitätsdimension abgeleitet. Die Tabelle 6.3 zeigt die jeweiligen Daten, die in ArEva als *QADAG*-Instanzen umgesetzt werden. In dem verfolgten Ansatz werden die Experimente somit als Betriebsmodi repräsentiert. Die Gewichtungen werden entlang der Anzahl und der Verhältnisse der Dimensionen festgelegt. Dabei werden vor allem ausbalancierte Dimensionen identisch gewichtet. Die Minimalwerte leiten sich analog zu den Datenblättern der Ressourcen ab, um die Auswahl spezifischer Ressourcen zu erzwingen. So bewirkt eine hohe Untergrenze für das Drehmoment, beispielsweise die Auswahl des Reaktionsradsystems.

Auf Grundlage der *QADAG*-Instanzen wird eine Abwägung der Qualitätsdimensionen für jeden Betriebsmodus vorgenommen und es werden drei Instanzen des *DARG* als betriebsoptimale Rekonfigurationsräume ausgebildet. Innerhalb dieser Graphinstanzen findet eine Priorisierung der Kanten, entsprechend der erreichbaren Nutzwerte von Alternativkonfigurationen, statt.

[16] Breite des Aufnahmestreifens auf der Erdoberfläche, gemessen in Kilometer.

Tabelle 6.3.: Gewichtungen und minimale Akzeptanzwerte pro Betriebsmodus

QADAG	noise	lifetime	pointingtime	accuracy	torque	powerconsumption
$N1$ Gewicht	0,2	0,175	0,125	0,125	0,175	0,2
$N1$ Min. Wert	0,2	0,45	0,0	0,0	0,0	0,1
$N7$ Gewicht	0,175	0,125	0,225	0,225	0,125	0,125
$N7$ Min. Wert	0,2	0,0	0,05	0,4	0,3	0,05
$N15$ Gewicht	0,075	0,2	0,175	0,175	0,2	0,175
$N15$ Min. Wert	0,0	0,35	0,0	0,35	0,3	0,15

Nach manueller Festlegung eines initialen Betriebsmodus und Startkonfiguration, wird das Modell parallel zum Betrieb ausgerollt. In der Alternativenexploration werden anschließend die Auswirkungen eintretender Fehler betrachtet.

6.6.3.2. Synthetische Fehlersequenzen

Als Teil der Systementwicklung werden umfangreiche Analysen zur Bestimmung von Ausfallraten und Eintrittswahrscheinlichkeiten für Fehlerfälle angestellt. Da das erweiterte ACS allerdings einen überdurchschnittlichen Grad an Redundanz und Austauschbarkeit besitzt, ist eine lange Sequenz von Fehlern notwendig, um das System kritisch zu beschädigen. Zur Präsentation des Ansatzes wurde eine *exemplarische Fehlersequenz* über die Hälfte der gesamten Ressourcenmenge ermittelt, die effektiv eine große Menge an Rekonfigurationen zulässt, vergleiche Tabelle 6.4.

Ohne Betrachtung einer spezifischen Konfigurationsmenge, ergibt sich die folgende Situation bei Beobachtung der Ressourcenplattform. Nach dem Durchlaufen der Fehlersequenz, sind in der Sensorik die Gyroskope und ein

Magnetfeldsensor weiterhin verfügbar. Mit dem 16. Fehler versagt dennoch die Fehlertoleranz des ACS, da die Aktuatorik zur Lageänderung ausfällt. Obwohl das Reaktionsradsystem mit drei Rädern grundsätzlich arbeiten kann, fehlt dazu das Magnetspulensystem zur Endsättigung. Entsprechend kann keine gültige Ressourcenkonstellation gefunden werden und ein Komplettversagen des ACS sowie der Verlust des Satellitens folgt.

Tabelle 6.4.: Exemplarische Fehlersequenz im ACS

#	Fehlerhafte Ressource	Redundanz	Ausfallerscheinung
1	GPS LNA1+Antenna1	OK	
2	GPS Receiver1	OK	
3	GPS LNA2+Antenna2	nicht-OK	ONS
4	MFS Fluxgate1	OK	
5	CSS RearHead2	OK	
6	ASC DPU1 High Res.	OK	
7	ASC DPU2 Low Res.	Nicht-OK	ASC
8	CSS Chipset1 High Res.	OK	
9	CSS RearHead1	Nicht-OK	CSS
10	CSS FrontHead2	Irrelevant	
11	CSS Chipset2 Low Res.	Irrelevant	
12	ASC DPU1 Low Res.	Irrelevant	
13	MC1x	OK	MSC_6_Coil
14	RW2	OK	
15	MC2y	OK	
16	MC2x	Nicht-OK	MSC und RW

Sobald eine konkrete Konfigurationsmenge betrachtet wird, ist nicht jede Konstellation der Ressourcen garantiert von einer Konfiguration abgedeckt. Dies bedeutet, dass potenziell nicht alle oben genannten Ressourcenausfälle behandelt werden können. Zur näheren Betrachtung ist die exemplarische Fehlersequenz auf die gesamte Ressourcenplattform auszuweiten.

Pivotierte Fehlersequenzen: Die exemplarische Fehlersequenz adressiert zwar einen relevanten Teil des Systems, allerdings kann das Eintreten und die Reihenfolge der Fehler nicht zuverlässig vorausgesagt werden. Um im

weiteren Verlauf belastbare Analyseergebnisse zu erheben, wird diese Fehlersequenz deshalb variiert. Eine kombinatorische Vertauschung der Fehler ist mit nicht annehmbaren Rechenaufwand möglich, da neben der Erzeugung der Sequenz vor allem die Erzeugung und Analyse des Rekonfigurationsraum für jede Variation wiederholt werden muss. Stattdessen wird eine teilweise Permutation der Fehler mit den insgesamt verfügbaren Ressourcen vorgenommen. Konkret wird die exemplarische Fehlersequenz schrittweise durchgegangen. An jeder Position (Index) wird die dort adressierte Ressource mit jeder anderen Ressource ausgetauscht, die bisher nicht Teil der Fehlersequenz war. Die restliche Fehlersequenz verbleibt unverändert. Daher handelt es sich bei der veränderten Ressource um ein Pivotelement. Die Tabelle 6.5 zeigt ein Beispiel für die Konstruktion der Pivotierung.

Für jeden Index werden alle verbleibenden Ressourcen einmalig eingesetzt. Aus den 32 Ressourcen verbleiben dabei auch 16 Optionen. Insgesamt werden somit $16 \times 16 = 256$ *pivotierte Fehlersequenzen* erzeugt. Zwar ist somit keine vollständige Abdeckung aller Fehlerkombinationen erfüllt, allerdings bedingt die Nähe zur exemplarischen Fehlersequenz einen hohen Grad an Rekonfigurierbarkeit.

6.6.3.3. Datenanalysen und Datenquellen

Die Analyse basiert auf einer betriebsoptiomalen Instanz eines *DARG*, unter Beachtung einer Obergrenze für Rekonfigurationskosten. In der anschließenden Analyse erfasst die Implementierung, entsprechend Abschnitt 5.2.4, graph- und pfadbasierte Kennzahlen. Diese Informationen werden in sieben Datenquellen für das Wartungspersonal aufbereitet und protokolliert.

Schwellwert für Rekonfigurationskosten: Die Kostengrenze begrenzt die maximalen strukturellen Änderungen bei Konfigurationswechseln. Der Anwender definiert die Obergrenze manuell. Im Folgenden wird für das ACS zunächst ein Viertel der Ressourcenmenge als Obergrenze angenommen, das heißt $\frac{32}{4} = 8$.

Tabelle 6.5.: Beispiel für pivotierte Fehlersequenzen im ACS

Permutation	Index 1	Index 2	Index 3	…	Index 16
Exemplar. Fehlerseq.	*GPS LNA1+Antenna1*	*GPS Receiver1*	*GPS LNA2+Antenna2*	…	*MC2x*
1.1	**GPS Receiver2**	GPS Receiver1	GPS LNA2+Antenna2	…	MC2x
1.2	**MFS Fluxgate2**	GPS Receiver1	GPS LNA2+Antenna2	…	MC2x
		…			
2.1	GPS LNA1+Antenna1	**GPS Receiver2**	GPS LNA2+Antenna2	…	MC2x
2.2	GPS LNA1+Antenna1	**MFS Fluxgate2**	GPS LNA2+Antenna2	…	MC2x
		…			
16.15	GPS LNA1+Antenna1	GPS Receiver1	GPS LNA2+Antenna2	…	**RW4**
16.16	GPS LNA1+Antenna1	GPS Receiver1	GPS LNA2+Antenna2	…	**CSS Chipset2 High Res.**

Graphbasierte Rekonfigurationsanalyse: In Unabhängigkeit zu einer Fehlersequenz werden Kennzahlen für den *DARG* extrahiert, zum Beispiel Graphdurchmesser oder die Kohäsion von Knoten. Weiterhin werden die Konfigurationen im Graphen paarweise verglichen um qualitative und strukturelle Unterschiede, ohne Beachtung von Kanten im Graphen, zu erfassen. Weiterhin werden die Ergebnisse der multi-kriteriellen Optimierung aufbereitet und normalisiert. Für jede Konfigurationen werden deren Ressourceneinsatz (individuell und als Anzahl) und der Nutzwert aufbereitet.

Pfadbasierte Rekonfigurationsanalyse: Jeder injizierter Fehler aus einer Fehlersequenz wirkt sich potenziell auf den Rekonfigurationsraum aus. Im Datenmodell entsteht ein *Rekonfigurationspfad* entlang der Abarbeitung der einzelnen Schritte in der Fehlersequenz. Jeder weitere Fehler erzeugt ein neues *Pfadsegment*, welches Informationen zur aktuell gewählten Architektur, der Fehlerinformation und diversen Kennzahlen zum Systemzustand bereithält. Tangiert ein Fehler die aktuell verwendete Konfiguration, wird eine Rekonfiguration angestoßen. Im Pfadsegment werden dann zusätzlich die Kennzahlen aus der Alternativensuche und die vorgeschlagene Zielkonfiguration protokolliert. Können keine Alternativen identifiziert werden, endet der Rekonfigurationspfad vorzeitig.

Tabelle 6.6.: Datenquellen zur Metrikendefinition

Datenquelle	Inhalte
1	Ressourcenanzahl und Nutzwert pro Konfiguration
2	Normalisierte Einzelqualitätswerte pro Konfiguration
3	Absolute Einzelqualitätswerte pro Konfiguration
4	Ressourcenauswahl pro Konfiguration
5	Qualitätsstreuung zwischen Konfigurationspaaren
6	Kostenstreuung zwischen Konfigurationspaaren
7	Rekonfigurationspfade entlang von Fehlersequenzen

Erzeugte Datenquellen: Zur Weiterverarbeitung stellt ArEva eine Reihe von Daten der graph- und pfadbasierten Analyse zur Verfügung, vergleiche Tabelle 6.6. Die Einordnung und Interpretation der erzeugten Daten,

vergleiche Abschnitt 6.7, erfordert eine Messung und Auswertung entlang der Validierungsfragen.

6.6.4. Metrikendefinition und Messungen

Die Implementierung des Ansatzes ermöglicht die Erhebung von Messdaten, im Rahmen eines simulativen Betriebs des Rekonfigurationsprozesses, zur Laufzeit. Auf Grundlage der variantenreichen Modellierung des Systems und den durchgeführten Analysen des Rekonfigurationsraums werden gemäß *GQM4* und *GQM5*, vergleiche Abschnitt 6.1.2, folgende Metriken definiert und ausgeführt. Zur Detailbetrachtung von Einzelfällen, beschreiben fehlermarkierte *DARG*-Instanzen (Pfadsegmente) konkrete Fehlersequenzen. Sämtliche Metriken werden für alle drei eingeführten Betriebsmodi durchgeführt, um die Aussagekraft der Ergebnisse zu steigern. Als initiale Konfiguration wird, wenn nicht anders beschrieben, eine Architektur angenommen, die sämtliche Ressourcen zur Ausführung nutzt.

6.6.4.1. Metrik für Validierungsfrage FF2.5

FF2.5 *„Ist der Kommunikationsaufwand im Betrieb durch frühzeitige Analysen des Entwurfsraums reduzierbar?“*

Folgende Aspekte betreffen die Messung der Kommunikationsaufwände.

Datencharakterisierung: Die in der Wartung zwischen System und Bodensegment übertragenen Daten quantisieren den betrachteten Kommunikationsaufwand. Zusätzlich zu den erzeugten Diagnosedaten des Systems ist auch die Datenmenge zur Fehlerbehebung relevant. Der Kommunikationsaufwand schwankt entsprechend der Signifikanz des Fehlers in Bezug auf die Ressourcenplattform. Bei redundanter Austauschbarkeit ist eine geringe Datenmenge zu erwarten; bei strukturell aufwendigen Rekonfigurationen werden hingegen viele Daten übertragen.

Datengrundlage: Der Ansatz verfolgt das Ziel notwendige Änderungen kompakt als abstrakte Rekonfigurationspfade zu beschreiben. Dabei wird auf die Vorausberechnung möglicher Konfigurationsalternativen zur Entwurfszeit zurückgegriffen. Diese Wissensbasis wird zur Laufzeit bei Betriebsmoduswechsel, vergleiche Abschnitt 5.4, oder Ressourcenausfällen kontinuierlich aktualisiert. In der Wartung erlaubt die Wissensbasis so im Bodensegment eine leichtgewichtige Erhebung nachgelagerter Ressourcen und Basiskomponenten zur Kompensation eines Ressourcenfehlers. In der Messung wird ein stufenweiser *Ausfall der Ressourcen RW3 und RW4* angenommen. Zur möglichst modusunabhängigen Beurteilung, werden die Messergebnisse gemittelt. Die Modusanzahl definiert Anforderungen an die Kapazität des Kommunikationskanals und ist daher Teil der Metrik.

Datenerhebung: Kommunikationsaufwände werden hier in übertragenden Bits[17] gemessen. In der Fallstudie liegen *32 Ressourcen* in der Ressourcenplattform vor, womit eine eindeutige *Ressourcen-ID maximal* 6 *Bits* beansprucht. Ausgehend von der graphbasierten Untersuchung der Konfigurationsräume der *DARG* Instanzen in Datenquelle 4, werden im Durchschnitt 112 Konfigurationsalternativen gemessen, wovon *32* ohne RW3 und RW4 ausführbar sind, vergleiche Tabelle 6.7.

Tabelle 6.7.: Konfigurationsalternativen ohne RW3 und RW4 für alle Betriebsmodi

Modus	Gesamt	Ohne RW3/4
N1	124	42
N7	104	22
N15	110	32
	112	**32**

Zur modusübergreifenden Abgrenzung der IDs für alle Pfadsegmente, wird der Datenbereich entsprechend mit der Modusanzahl multipliziert. Skaliert auf die drei Betriebsmodi in der Fallstudie, resultiert eine Verdreifachung der maximalen Alternativenanzahl ($3 \times 32 = 96$ eindeutige Segmente), was

[17] entspricht der Länge der jeweiligen Identifikationsnummern (*ID*) für Ressourcen und Pfadsegmente

sich in *Segment-IDs der maximalen Länge von* 6 *Bits* abbilden lässt. Die Analyse verläuft *graphbasiert* und ist unabhängig von einem konkreten Rekonfigurationspfad. Die Approximation basiert auf der Ressourcenanzahl, der Modusanzahl und den möglichen Rekonfigurationsoptionen.

Zur Aktivierung der jeweiligen Kanten im deterministischen Architekturrelationsgraphen, wird ausschließlich eine ID der ausgefallenen Ressource vom Bodensegment zum Satelliten übertragen. Durch die Vorausberechnung des Rekonfigurationsraums und dessen kontinuierlicher Aktualisierung, bedarf auch die Datenübertragung in Gegenrichtung nur der Übermittlung der ID des gewählten Pfadsegments. Ein zusätzliches Bit bestätigt (1) oder widerlegt (0) die Aktivierung der neuen Konfiguration. Von möglichen zusätzlichen Daten für Informationsredundanzen wird hier abgesehen.

Ergebnis: Es entstehen zur Laufzeit im Fehlerfall die nachfolgend aufgeführten maximalen Kommunikationsaufwände. Ein Rechtspfeil beschreibt den Datenfluss vom Satelliten zum Bodensegment; ein Linkspfeil die Gegenrichtung.

	6	b	→ ID fehlerhafte Ressource
+	6	b	← ID gewählte Pfadsequenz
+	7	b	→ Aktivierungsbit + Wiederholung der Pfadsequenz-ID
	19	b	Gesamte Datenmenge

Im Rahmen der Validierung des Kommunikationsaufwands, unter Verwendung des *DARG*-Ansatzes, entsteht eine maximale theoretische Datenmenge unter 3 *Bytes* zur Übertragung zwischen Satelliten und Bodensegment, wobei weitere fünf Bits ungenutzt bleiben. Eine Einschränkung dieser Art der Kommunikation besteht darin, dass die gewählte Sequenz und die zugehörige Rekonfigurationsstrategie im Produktivsystem bereits implementiert sein müssen. Dies sollte jedoch für alle konkreten Rekonfigurationspfade die auf Grundlage der erwarteten Fehler, vergleiche Abschnitt 6.6.3.2, zur Entwurfszeit des fehlertoleranten Systems vorberechnet wurden gelten.

6.6.4.2. Metrik für Validierungsfrage FF2.6

FF2.6 *„Führt eine Priorisierung von Rekonfigurationsentscheidungen zu einer Reduktion des personellen Wartungsaufwands?“*

Die Messung der personellen Aufwände im Bodensegment in der Wartung lassen sich wie folgt erheben.

Datencharakterisierung: Unabhängig von der Vorbereitung einer Strategie zur Fehlerbehandlung bleibt die Wartung im Betrieb, aufgrund vieler atomarer Aktionen, personell grundsätzlich aufwendig. So muss das Wartungspersonal neben der Ursache eines Fehlers, dessen Einfluss auf die laufende Systemkonfiguration beurteilen, Alternativen bezüglich deren Anwendbarkeit abwägen und letztlich die Systemänderung umsetzen. Insbesondere die Aktivierung beziehungsweise Deaktivierung betroffener Ressourcen und kausal verknüpften Basiskomponenten in der Software entlang eines Rekonfigurationspfads, beeinflussen die durchschnittliche Wartungszeit[18] maßgeblich.

Datengrundlage: Der Personalaufwand wird anhand der Komplexität von Rekonfigurationsentscheidungen beurteilt. Diese Daten lassen sich aus den erhobenen Rekonfigurationspfaden ableiten und bezüglich der qualitativen und strukturellen Veränderungen des Systems untersuchen. Die Approximation der qualitativen Bewertungen jeder Architektur zur Entwurfszeit ermöglicht eine Priorisierung der Alternativensuche zur Laufzeit. Aus einem qualitätspriorisierten Vergleichsverfahren wählt der Ansatz die qualitativ beste Alternativen automatisiert aus. Die ungeführten Vergleiche betrachten alle verfügbaren und direkt verbundenen Architekturen. Dabei werden insbesondere auch diejenigen Architekturen betrachtet, die qualitativ eine ungenügende Bewertung gemäß des Qualitätsattributsgraphens erhalten haben. Diese Architekturen entsprechen nicht den Betriebsanforderungen und dienen ausschließlich als Rückfallebene. Zugleich werden die Rekonfigurationskosten zur Limitierung der Suchtiefe beachtet. Die Minimierung des deterministischen Architekturrelationsgraphens beinhaltet, dass zur qualitativen Abwägung nur strukturell ähnliche Konfigurationen zur Auswahl stehen.

[18] Im Sinne der MTTR (Englisch: mean time to repair).

Datenerhebung: Die Gesamtlänge eines Rekonfigurationspfads entspricht der Anzahl der Pfadsegmente, die vom Personal begutachtet werden müssen, um das System wieder in einen operativen Zustand zu versetzen. In jedem Segment sind unterschiedliche Alternativen zu vergleichen. Beide Faktoren sind somit Gegenstand der Messung. Verursacht ein Ressourcenfehler keinerlei Schäden an der aktuellen Konfiguration, kommt es auch zu keiner Rekonfiguration. Quell- und Zielarchitektur sind in diesem Fall im Segment identisch. De facto kommt es also zu keinem Vergleich. Die Messung basiert auf der pfadbasierten Analyse aus Datenquelle 7 und bezieht sich demnach auf konkrete Rekonfigurationspfade entlang einer Fehlersequenz. Die Rekonfigurationskostenschwelle ist auf 8 Änderungen gesetzt. Alle drei Betriebsmodi werden in Kombination mit der exemplarischen Fehlersequenz aus Abschnitt 6.6.3.2 betrachtet. Die Fehler werden in der beschriebenen Reihenfolge injiziert und erzeugen pro Vorgang ein neues Segment im Rekonfigurationspfad. Sobald ein Fehler die genutzten Ressourcen der aktuellen Konfiguration tangiert, wird eine Alternativensuche im *DARG* angestoßen. Der Prozess terminiert, sobald die Exploration keine weitere gültige Alternative identifiziert oder sämtliche Fehler abgearbeitet wurden.

Ergebnis: Für jeden Rekonfigurationsschritt wird die Anzahl der zu vergleichenden Alternativen gemessen, die (1) aktuell erreichbar, (2) unmittelbar verbunden oder (3) qualitativ priorisiert sind. Abbildung 6.12 zeigt die aufsummierten Vergleiche pro Betriebsmodus.

Die Erreichbarkeit ist mit durchschnittlich *79* nahezu identisch über alle Betriebsmodi hinweg gegeben. Die priorisierten Vergleiche dominieren insbesondere bei $N7$ die ungeführten Vergleiche. Durchschnittlich ergibt sich ein Verhältnis von rund *16* ungeführten zu *10* priorisierten Vergleichen.

Weiterhin wird bei jeder erfolgten Rekonfiguration die Länge des Pfads festgehalten. Abstände >1 im Pfad bedeuten, dass Fehler ohne Rekonfiguration kompensiert wurden (Redundanzen) oder nicht relevant sind für die aktuelle Konfiguration. Abschließend werden die Rekonfigurationskosten gemessen, das heißt die (De)aktivierungen von Ressourcen, die beim Konfigurationswechsel auftreten. In Tabelle 6.8 sind die zusammengefassten Ergebnisse der Messung abgetragen.

Tabelle 6.8.: Vergleiche und Kosten in der Alternativensuche für alle Betriebsmodi

Rekonfiguration	Quelle	Ziel	Erreichbar	Ungeführt	Priorisiert	Pfadlänge	Kosten
N1.1	4701	4470	22	3	3	3	8
N1.2	4470	4745	22	1	1	9	8
N1.3	4745	4316	17	5	4	13	4
N1.4	4316	4406	17	3	2	14	2
	4701	**4406**	**78**	**12**	**10**	**16**	**22**
N7.1	4567	4338	20	3	3	3	8
N7.2	4338	4401	20	6	2	12	8
N7.3	4401	4381	20	6	2	13	4
N7.4	4381	4362	20	4	1	14	2
	4567	**4362**	**80**	**19**	**8**	**16**	**22**
N15.1	4726	4450	22	1	1	9	8
N15.2	4450	4401	20	6	4	12	6
N15.3	4401	4316	20	6	4	13	8
N15.4	4316	4406	17	3	2	14	2
	4726	**4406**	**79**	**16**	**11**	**16**	**24**
			79,0	**15,7**	**9,7**	**16,0**	**22,7**

Der Ansatz ermittelt für alle Betriebsmodi und die betrachtete Fehlersequenz jeweils *vier Rekonfigurationen*. Die Pfadlänge entspricht für jeden Betriebsmodus jeweils *16*, das heißt dass alle Fehler vom System abgefangen werden können. Dies trifft auch auf den Ausfall kompletter Redundanzgruppen zu. Die Zusammensetzung der Pfade ist sehr unterschiedlich. Die initiale Konfiguration 4726 von $N15$ fängt 9 Fehler ab, $N7$ verbleibt dagegen für 9 Schritte in 4338. Diese Konfigurationen zeichnen sich durch eine hohe Robustheit aus und können sich zugleich qualitativ gegenüber der Alternativkonfigurationen behaupten. Die Rekonfigurationskosten liegen im Schnitt bei circa 23 Änderungen und sind nahezu identisch für alle Betriebsmodi. So

ergeben sich durchschnittlich rund 6 pro Rekonfiguration. Die Rekonfigurationskostenschwelle wird in der Hälfte aller Rekonfigurationen ausgereizt. Ansonsten liegen die Kosten unterhalb von 50% der Obergrenze. Ein Zusammenhang zwischen der Robustheit und den resultierenden Kosten beim Konfigurationswechsel ist nicht erkennbar. Zwar reizen die Wechsel der genannten zwei Konfigurationen die Grenze zwar auch, dies kommt aber auch bei weniger robusten Konfigurationen vor, zum Beispiel 4401 bei $N15$. Die jeweils letzte Rekonfiguration verursacht modusunabhängig geringe Kosten. Die ist dadurch zu erklären, dass bei den Pfadlängen von jeweils 14, der Rekonfigurationsraum schon sehr ausgedünnt ist und die Ähnlichkeit der gebundenen Ressourcen zwischen Konfigurationen zunimmt.

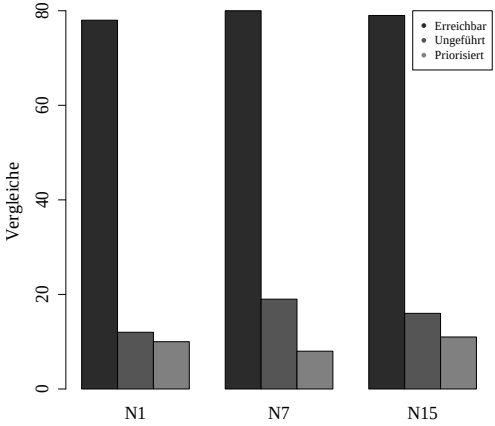


Abbildung 6.12.: Vergleiche in der Alternativensuche für alle Betriebsmodi

6.6.4.3. Metrik für Validierungsfrage FF3.1

FF3.1 *„Ist eine kompakte visuelle Darstellung des Rekonfigurationsraums umsetzbar?“*

Die quantitative Bewertung einer reichhaltigen und zugleich übersichtlichen Darstellung des Rekonfigurationsraums in Form eines deterministischen Architekturrelationsgraphens erfolgt im Anschluss.

Datencharakterisierung: Der Rekonfigurationsraum spannt den Entscheidungsraums mit einer Vielzahl möglicher alternativer Lösungen auf. Alle Konfigurationsalternativen stehen sowohl strukturell als auch qualitativ in fernen beziehungsweise nahen Beziehungen. Die Untersuchung von möglichen Folgekonfigurationen und damit verbundenen Nebenwirkungen stellt hohe Anforderungen an die Übersichtlichkeit und Nachvollziehbarkeit der Rationalen für Rekonfigurationsvorschlägen.

Datengrundlage: Neben der Größe des Rekonfigurationsraums, quantisiert durch verbleibende valide Konfigurationen, ist zur Beurteilung der Rekonfigurierbarkeit ebenfalls die Raumdichte relevant. Entsprechend Abschnitt 4.1.2 werden Konfigurationen durch Knoten repräsentiert, Beziehungen durch Kanten. Die Größe des Suchraums entspricht der Knotenmenge. Die Dichte entspricht dem Verhältnis zwischen Kanten im minimierten *DARG* und (nicht-minimiertem) *DARG*. Zusätzlich wird der Durchmesser je Graph bestimmt. Diese entspricht dem jeweils kürzesten Pfad zwischen zwei maximal weit entfernten Knoten. Bezüglich der graphübergreifenden Nachvollziehbarkeit ist ein geringer Durchmesser dabei negativ zu bewerten, da dieser auf eine intensive Vermaschung des Graphens hinweist. Zusätzlich werden die Rekonfigurationskosten und qualitative Distanzen zwischen allen Konfigurationspaaren bestimmt und gemittelt. Dies ermöglicht eine Beurteilung der durchschnittlichen Kohäsion zwischen validen Konfigurationen und somit eine effektive Bemessung der Kompaktheit des deterministischen Architekturrelationsgraphens.

Datenerhebung: Die genannten Kennzahlen der erzeugten deterministischen Architekturrelationsgraphen sind rein objektive Messgrößen, die jedoch Rückschlüsse auf die subjektiv wahrgenommene Transparenz der vorgeschlagenen Systemmodifikation durch das Wartungspersonal zulassen. Die Graphkomplexität schwankt abhängig von dem Schwellwert für die maximalen Rekonfigurationskosten, da dieser den Minimierungsgrad beeinflusst. Zur repräsentativen Beurteilung betrachtet die Messung mehrere minimierte deterministischen Architekturrelationsgraphen in Reihe. Ausgehend von der gesamten Ressourcenmenge, wird eine Spanne von ein bis sieben Achtel der maximal 32 möglichen Ressourcenanpassungen untersucht, das heißt Grenzen von 4, 8, 12, 16, 20, 24 und 28. Zusätzlich werden die Daten ohne

Minimierung (∞) ausgewertet, was einer Grenze von 32 entspricht. Jeweils eine Instanz der Datenquelle 7 beinhaltet die Messdaten pro Minimierung im initialen Zustand der Rekonfigurationspfade.

Ergebnis: Insgesamt werden sieben minimierte deterministische Architekturrelationsgraphen und der vollständige Graph pro Betriebsmodus abgeleitet, vergleiche Tabelle B.3 in Anhang B. Im Folgenden werden die Durchschnittswerte über alle Betriebsmodi angegeben. Die Anzahl der Knoten liegt bei 112 für alle Minimierungen, da die Minimierung ausschließlich Kanten reduziert. Die Kantenanzahl der betrachteten Minimierungen belaufen sich auf rund 250, 1400, 3100, 5600, 8600, 10800 und 12000 für alle validen Konfigurationsalternativen. Für die vollständigen Graphen resultieren rund 12700 Kanten. Dieser Wert stimmt mit dem rechnerischen Wert nach $2 * \binom{n}{2} = \frac{2n(n-1)}{2}$ für n Knoten in einem gerichteten Graphen überein. Je nach Grad der Minimierung variiert die Kantenanzahl um den Faktor 50.

Zur Bestimmung der Dichte der Graphen, ist die jeweilige maximale Kantenanzahl der vollständigen Graphen mit denen der minimierten Graphen in Verhältnis zu setzen. Folglich ergibt sich als Dichte der 4er-Minimierung $\frac{250\ Kanten}{12700\ Kanten} = 0,02$. Für die weiteren Zähler ergeben sich Dichten von 0,11, 0,24, 0.44, 0.68, 0.85 und 0,94. Eine geringe Dichte weist zugleich darauf hin, dass der Graph weniger stark vermascht ist. Dies fördert wiederum die Übersichtlichkeit. Dabei ist es allerdings weiterhin relevant, dass neben Paaren auch längere Pfade entstehen.

Die Graphdurchmesser sind in Abbildung 6.13 für alle Minimierungen und die drei Betriebsmodi abgetragen. Für die Messreihe liegt der Durchmesser vorrangig zwischen 3 und 4 Kanten. Für jeden Betriebsmodus sticht die Minimierung auf maximal 8 Ressourcenänderungen allerdings heraus. Dort treten Werte von 6 und 8 Kanten auf. Dies weist darauf hin, dass eine Minimierung von 8 Einheiten die transiente Konfigurationen optimal ausnutzt und zugleich eine geringe Dichte aufweist. Im Verhältnis zum vollständigen Graphen, resultiert eine deutliche Steigerung der Übersichtlichkeit von 89%, ohne das hohe Kostengrenzen festgelegt werden müssen.

Eine kompakte Darstellung der Rekonfigurationsräume ist durch die Kantenminimierung möglich, geht aber mit einer Einschränkung der Rekonfigurierbarkeit einher. Durch das Entfernen von Kanten werden transiente

Übergänge zwischen Knoten notwendig. Zugleich steigt durch das Streichen redundanter Pfade die Graphtransparenz. Hier orientiert sich der Ansatz jedoch daran, dass Konfigurationen mit ähnlichen Ressourcen dicht zusammen liegen. Im Sinne der kostenorientierten Wartung, wird aus dieser Menge von Alternativen sinnvollerweise gewählt.

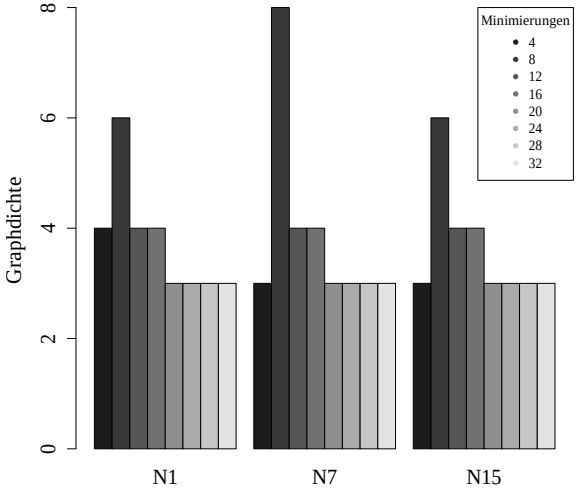


Abbildung 6.13.: Graphdurchmesser bei variablen Rekonfigurationskostengrenzen für alle Betriebsmodi

6.6.4.4. Metrik für Validierungsfrage FF3.2

FF3.2 *„Sind Zusammenhänge zwischen Fehlereintritt und Fehlerauswirkung nachvollziehbar darstellbar?“*

Eine abstrakte und zugleich nachvollziehbare Darstellbarkeit der Zusammenhänge zwischen Eintritt und Auswirkung eines Ressourcenfehlers, unter Einsatz des vorgestellten Ansatzes, wird im Anschluss untersucht.

Datencharakterisierung: Die Fehlerauswirkung bildet sich als Unterschied zwischen momentaner Architektur und der ermittelten Austauscharchitektur

aus; quantitativ ausgedrückt als Aufwand zur Anpassung der Ressourcen und Softwarearchitektur. Mit der Reihenfolge der Fehler variiert auch die Anzahl möglicher Rekonfigurationen und die Länge der Rekonfigurationsschritte. Dies korreliert mit der Intensität der Auswirkung eines Fehlers.

Datengrundlage: Die pfadbasierte Messung erfolgt durch eine Begutachtung der Rekonfigurationspfade aus Datenquelle 7 für die pivotierten Fehlersequenzen aus Abschnitt 6.6.3.2. Auf Basis der vorherigen Untersuchungen wird die Minimierung auf maximal 8 Ressourcenänderungen festgesetzt. Der Eintritt eines Fehlers stimuliert im deterministischen Architekturrelationsgraphen die Auswahl eines adäquaten Rekonfigurationspfads, sofern dieser existiert. Diese Pfadauswahl wird in dem vorgestellten Verfahren durch qualitative und strukturelle Vorgaben gesteuert. Dem Anwender wird ein eindeutiger und zyklenfreier Pfad präsentiert, vergleiche Abschnitt 5.2.3. Treten mehrere Fehler in einer Sequenz auf, wird der Pfad kontinuierlich verlängert, solange gültige Alternativen existieren. Die Anzahl der traversierten Kanten, sichtbar als Rekonfigurationsaufwand, entlang möglicher Fehlersequenzen beeinflussen die Nachvollziehbarkeit der Rekonfigurationshistorie maßgeblich. Die Entwicklung entlang eines Pfads werden über alle Fehlersequenzen aus Abschnitt 6.6.3.2 untersucht. Je nach Fehlerreihenfolge resultieren individuelle Rekonfigurationspfade mit Kennzahlen. Zur Einordnung der Ergebnisse werden die Fehlerauswirkungen innerhalb von Ober- und Untergrenzen eingeordnet.

Datenerhebung: Die pfadbasierte Datenerhebung greift zunächst auf die exemplarische Fehlersequenz aus Abschnitt 6.6.3.2 für alle Betriebsmodi zurück. Hierbei werden die Länge, Rekonfigurationsmenge und -kosten aus Abschnitt 6.6.4.2 aufgegriffen. Über alle pivotierten Fehlersequenzen werden anschließend die minimalen und maximalen Kennzahlen gemessen. So wird die jeweils optimale sowie kritische Fehlereintrittsreihenfolge erhoben. Weiterhin wird eine visuelle Darstellung des vorgeschlagenen Rekonfigurationspfads angeboten.

Ergebnis: Entlang der exemplarischen Fehlersequenz ergeben sich für jedes Segment zunächst die Kennzahlen entsprechend der Durchschnittswerte aus Tabelle 6.8 zur Messung in Abschnitt 6.6.4.2. Aus der Menge der 257

Permutationen der Fehlersequenz werden Ober- und Untergrenzen für die Länge der Rekonfigurationspfade sowie der Anzahl und Kosten erfolgter Rekonfigurationen berechnet. Tabelle 6.9 zeigt die Ergebnisse der Messung. Die Werte der exemplarischen Fehlersequenz sind für jedes Tripel zuerst aufgeführt. Die jeweils zu vergleichenden Werte sind fett gedruckt.

Die ermittelten optimalen Fehlersequenzen sind in Tabelle B.4 in Anhang B aufgeführt. Die exemplarische Fehlersequenz (Permutation 1) nimmt in dieser Untersuchung die maximale Länge von 16 Schritten für alle Betriebsmodi ein. Zusätzlich führt sie für Betriebsmodus $N7$ auch zur maximalen Anzahl der Rekonfigurationen und gilt dabei als optimale Sequenz. Für die Betriebsmodi $N1$ und $N15$ sind maximal 5 Rekonfigurationen mit anderen Permutationen möglich. Bei $N1$ führt die Permutation 59 dabei zwar zu 5 Rekonfigurationen und ist damit ein Kandidat als optimale Sequenz, allerdings arbeitete sie nur 15 Fehler ab. Die Permutation 1 akzeptiert hingegen 16 Fehler. So ist für $N1$ keine eindeutige Ermittlung einer optimalen Fehlersequenz möglich. Analog gilt dies für $N15$ wobei die Permutation 33 die exemplarische Fehlersequenz dort in der Anzahl dominiert.

Bei der Identifikation der kritischen Fehlersequenz, siehe Tabelle B.5 in Anhang B, fällt die Permutation 3 auf, die bei Betriebsmodus $N15$ sowohl zu einer minimalen Länge (9) als auch Anzahl (1) führt. Das globale Längenminimum erzeugt die Permutation 27, die bei $N7$ nach nur drei Schritten zum Totalausfall des Systems führt. Wie auch Permutation 27 erreicht diese Fehlersequenz nur eine Rekonfiguration und kann daher als kritisch für $N7$ angesehen werden. Für Betriebsmodus $N1$ ist keine eindeutige kritische Fehlersequenz zu ermitteln, da die Permutationen 22 und 3 sich jeweils einseitig ($9 < 16$ bzw. $2 > 1$) dominieren.

In der Fallstudie sind die individuellen Kosten für das Zu- und Abwählen der Ressourcen identisch und koorelieren somit mit der Anzahl der Konfigurationsänderungen. Daher erfolgt die Auswahl einer optimalen und kritischen Fehlersequenz pro Betriebsmodus nicht entlang dieser Werte. Die Abbildung 6.14 zeigt die Verteilung der Kosten über alle Fehlersequenzen hinweg. Dabei sind die Werte der exemplarischen Sequenz hervorgehoben.

Für den Betriebsmodus $N1$ entsprechen die Kosten dem Median. Gemessen an den dennoch vier durchgeführten Rekonfigurationen halten sich die Kosten deutlich im Rahmen im Vergleich zu den maximal möglichen fünf Rekonfigurationen. Bei $N7$ entsprechen die Kosten dem oberen „Whisker“.

Tabelle 6.9.: Pfadbasierte Kennzahlen über Fehlersequenzen für alle Betriebsmodi

Modus	Permutation	Kennwert	Länge	Anzahl	Kosten
N1	1	Pfadlänge	**16**	4	22
N1	22	Minimum	**9**	2	16
N1	1	Maximum	**16**	4	22
N1	1	Anzahl	16	**4**	22
N1	3	Minimum	16	**1**	8
N1	59	Maximum	15	**5**	34
N1	1	Kosten	16	4	**22**
N1	3	Minimum	16	1	**8**
N1	59	Maximum	15	5	**34**
N7	1	Pfadlänge	**16**	4	22
N7	27	Minimum	**3**	1	8
N7	1	Maximum	**16**	4	22
N7	1	Anzahl	16	**4**	22
N7	19	Minimum	4	**1**	8
N7	1	Maximum	16	**4**	22
N7	1	Kosten	16	4	**22**
N7	19	Minimum	4	1	**8**
N7	1	Maximum	16	4	**22**
N15	1	Pfadlänge	**16**	4	24
N15	3	Minimum	**9**	1	4
N15	1	Maximum	**16**	4	24
N15	1	Anzahl	16	**4**	24
N15	3	Minimum	9	**1**	4
N15	33	Maximum	16	**5**	30
N15	1	Kosten	16	4	**24**
N15	3	Minimum	9	1	**4**
N15	33	Maximum	16	5	**30**

Hier resultiert die Durchführung das Maximums an möglichen Rekonfigurationen auch in maximalen Kosten. Bei Betriebsmodus $N15$ liegen die Kosten der exemplarischen Sequenz im oberen Quartil. Analog zu $N1$ treten hier

vier von fünf möglichen Rekonfigurationen auf, allerdings ist die Effizienz des Rekonfigurationspfads schlechter.

Da der Anwender nur eingeschränkt Einfluss auf die Reihenfolge der Fehler hat, ist allerdings von der schlecht-möglichsten Rekonfigurierbarkeit auszugehen, welche zugleich diese kritischste Fehlersequenz beschreibt.

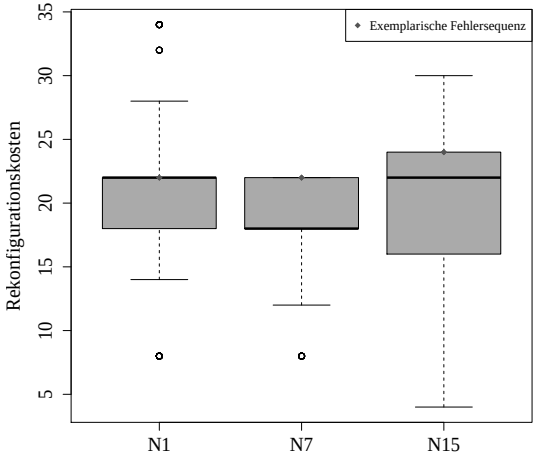


Abbildung 6.14.: Kostenverteilung über Fehlersequenzen für alle Betriebsmodi

Visuell betrachtet resultiert nach dem Durchlaufen der exemplarischen (hier: optimalen) Sequenz für den Betriebsmodus $N7$ der in Abbildung 6.15 gezeigte Rekonfigurationspfad[19].

Die Rekonfiguration entlang der Fehlersequenz beginnt in Konfiguration 4567 und endet in Konfiguration 4362. Einige Konfigurationen kompensieren die auftretenden Fehler[20] und es findet kein Konfigurationswechsel statt. Im Rekonfigurationspfad steigt dann zwar die Pfadlänge[21], die Konfiguration bleibt aber identisch.

[19] Die Knoten sind entsprechend Abschnitt 5.2.3 farblich kodiert; hier indiziert durch b (blau), g (grün) und r (rot).
[20] An den Kanten annotiert
[21] Dargestellt als geklammerte Zahl vor der Architektur-ID

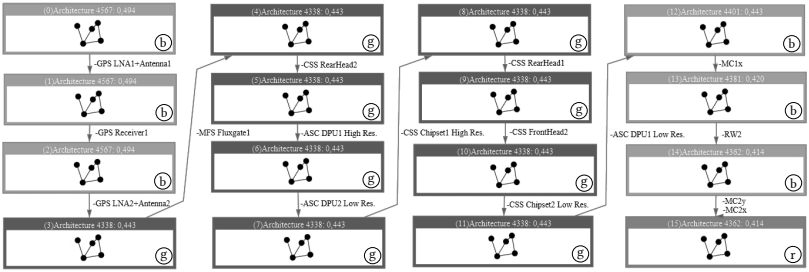


Abbildung 6.15.: Rekonfigurationspfad nach exemplarischer Fehlersequenz des Betriebsmodus N7

Die gezeigte Variante der Visualisierung verzichtet auf das Anzeigen aller möglichen Alternativen, vergleiche Metrik 2.6 in Abschnitt 6.6.4.2. Stattdessen werden nur die qualitativ hochwertigen[22] Alternativen entlang des Pfads dargestellt. Weitere alternative Konfigurationen sind im zugrundeliegenden Datenmodell zu jedem Pfadsegment abgelegt und können optional zusätzlich dargestellt werden. Der Pfad endet mit den Fehlern $MC2y$ und $MC2x$, welche nicht kompensiert werden können.

6.6.4.5. Metrik für Validierungsfrage FF4.1

FF4.1 *„Ermittelt ein metaheuristisches Suchverfahren einen relevanten Ausschnitt des alternativen Entwurfsraums?“*

Nachfolgend wird die Effektivität des meta-heuristischen Suchverfahrens bei Bestimmung einer adäquaten Alternativkonfiguration im Entwurfsraum untersucht.

Datencharakterisierung: Die Effektivität des Verfahrens lässt sich in Form einer Beurteilung der kostenorientierten Minimierung eines deterministischen Architekturrelationsgraphens bemessen. Werden dabei ausschließlich

[22] Nutzwerte hinter der Architektur-ID; Pareto-effiziente Konfiguration sind grün (g) eingefärbt

Kanten zu qualitativ nachrangigen Alternativen gestrichen, verbleiben letztendlich auch nur modusrelevante Konfigurationen in der Lösungsmenge. Die nachgelagerte explorative Analyse dieser Menge fokussiert daher die Eingrenzung auf qualitativ hochwertige Alternativen.

Datengrundlage: Die Anzahl der Kanten in einem *DARG* wird durch die Festlegung einer Obergrenze für Rekonfigurationskosten minimiert, vergleiche Abschnitt 5.2.2. So resultiert ein kostenminimierter *DARG* (hier: 8 Kosteneinheiten) in dem unverbundene Knoten zusätzlich entfallen. Zur Bewertung der Minimierung wird exemplarisch der Vergleich zur Suche im vollständigen Rekonfigurationsraum[23] ohne Kostenobergrenze für alle Betriebsmodi und die exemplarische Fehlersequenz durchgeführt[24]. Da nach Konstruktion der Minimierung jede relevante Alternative von einer beliebigen Konfiguration aus weiterhin erreichbar sein soll, fokussiert sich die Messung auf die Menge valider Konfigurationen für eine spezifische Ressourcenmenge. So lassen sich abhängig von dem aktuell ausgewählten Knoten und dessen Vernetzung, die Alternativen in beiden Graphvarianten abwägen.

Datenerhebung: Im vollständigen *DARG* sowie im minimierten *DARG* sind die jeweils eingesetzten Ressourcen pro Konfiguration annotiert. In der Messung werden diese Informationen über alle drei Betriebsmodi in Verhältnis gesetzt, um die Kennzahlen zu vergleichen. Zunächst sind die Varianten der Graphen auf die Anzahl möglicher Alternativen zu untersuchen. Hierfür werden die Vergleiche in der Alternativensuchen herangezogen. Abweichend von der Metrik 2.6 aus Abschnitt 6.6.4.2 werden dabei Durchschnitte gebildet, um die tatsächlich vorhandenen Alternativen pro Betriebsmodus zu ermitteln. Da das Suchverfahren grundsätzlich qualitätsorientiert vorgeht, werden die identifizierten Alternativen entsprechend priorisiert im *DARG* angeordnet. Dabei werden im vollständigen Graphen Rekonfigurationskosten nicht

[23] Es handelt sich hierbei ausdrücklich nicht um den kombinatorisch möglichen Entwurfsraum, sondern weiterhin um eine vorgelagerte Selektion aus diesem, vergleiche Abschnitt 3.

[24] Die Suche im vollständigen *DARG* ist nach dem Prinzip „brute-force“möglich, da der Entwurfsraum der Fallstudie dies zulässt. Im Allgemeinen ist eine erschöpfende Suche unter annehmbarem Aufwand nicht möglich. Hier müsste auf Expertenwissen zur sinnvollen Vorauswahl zurückgriffen werden.

behandelt. Somit sind die qualitativen und strukturellen Differenzen innerhalb beiden Varianten des deterministischen Architekturrelationsgraphens von Interesse. Die Messdaten für den minimierten *DARG* basieren auf der Metrik 2.6 in Abschnitt 6.6.4.2 mit einer zusätzlichen Messung der qualitativen Unterschiede zwischen Konfigurationen. Die weitere Messung im vollständig deterministischen Architekturrelationsgraphen soll zeigen, dass qualitativ relevante Alternativen auch nach der Minimierung mehrheitlich erhalten bleiben, das heißt keine Verbindungen zu diesen Knoten entfernt werden. Existiert eine hohe Anzahl identischer oder vergleichbarer Alternativkonfigurationen in beiden Graphvarianten, ist die Effektivität hoch; entfallen hingegen vorteilhafte Alternativen mit der Minimierung, ist das Verfahren potenziell ineffektiv.

Ergebnis: Die Anzahl der möglichen Konfigurationen pro Betriebsmodus und Graphvariante sind im Balkendiagramm in Abbildung 6.16 abgetragen. Für jeden Modus werden die Kennzahlen aus beiden Graphen verglichen.

In dem vollständigen Graphen sind grundsätzlich deutlich mehr Konfigurationen für jede Kategorie erreichbar. Dabei fällt auf, dass die Betriebsmodi $N1$ und $N7$ nahezu identische Werte haben. Modus $N15$ weicht durch einen anderen Ressourceneinsatz davon ab. Für die ersten beiden Modi ergibt sich ein Minimierungsgrad von circa 90%; Bei der ausschließlichen Betrachtung der Erreichbarkeit, wird für Betriebsmodus $N15$ ein ähnlicher Wert (circa 88%) erreicht. Über alle Kriterien hinweg enthält der minimierte *DARG* für $N1$ auch circa 70% der Alternativen.

Die Zahlen zeigen auf, dass die Minimierung für 8 Kosteneinheiten einen starken Einfluss auf die Anzahl der Alternativen hat.

Diesbezüglich wird im Folgenden überprüft, ob vorrangig irrelevante Alternativen verlagert wurden. Die Konfigurationen pro Graphvariante sind mit auftretenden Differenzen von Rekonfigurationskosten und Qualitäten[25] in Tabelle 6.10 aufgeführt. In der linken Hälfte sind die Konfigurationen für den vollständigen *DARG* abgetragen, rechts die für den minimierten. Für eine bessere Präsentation wurden leere Zellen hinzugefügt, um Gemeinsamkeiten von Quell- und Zielkonfigurationen darzustellen.

[25] Die Werte vom minimierten Graphen werden von denen des vollständigen Graphens subtrahiert.

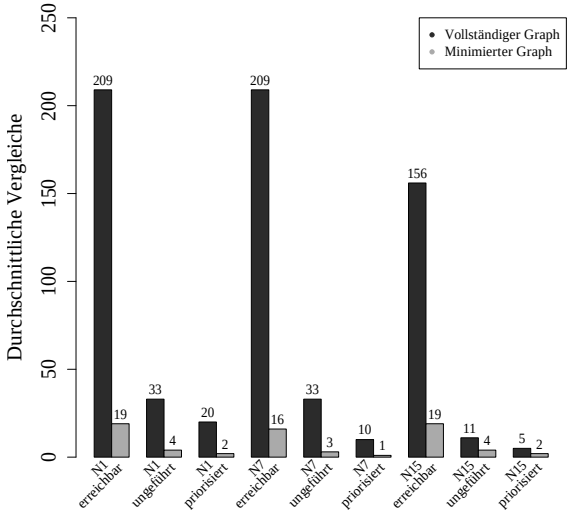


Abbildung 6.16.: Vergleiche in der Alternativensuche der Varianten des Rekonfigurationsraums für alle Betriebsmodi

Bei dem Betriebsmodus $N1$ werden auch im vollständigen *DARG* maximal vier Rekonfigurationen durchgeführt. Allerdings wird die die Konfiguration 4519 anstatt 4470 auf Grund eines knapp 3-prozentigen Qualitätsvorteils gewählt. Dieser Vorteil wird mit der nächsten Rekonfiguration allerdings wieder verloren. Dieser alternative Pfad im Graphen führt zu insgesamt 12 zusätzlichen Änderungen und steigert damit die Rekonfigurationskosten.

Im Gegensatz zur der Suche im minimierten Graphen, kann für Modus $N7$ die Anzahl der Rekonfigurationen auf fünf gesteigert werden. Dies weißt zwar auf einen größeren Alternativenraum hin, allerdings auch zu enormen Rekonfigurationkosten in der Höhe von 36 Einheiten für die An- und Abwahl der zusätzlichen Konfiguration. Obwohl zunächst die Qualität der Alternative circa 5% besser ist, kommt es in Summe zu keinem qualitativen Vorteil.

Für $N15$ wird im vollständigen *DARG* die Konfiguration 4745 anstatt 4450 und 4401 ausgewählt. Dies führt zwar zu einer Verringerung der Anzahl der Rekonfigurationen, steigert aber die Qualität um circa 3%. Dieser Vorteil wird

auf dem Pfad zurück zur Konfiguration 4316 jedoch wieder verloren, wobei die zusätzlichen Rekonfigurationskosten allerdings kompensiert werden.

Tabelle 6.10.: Kennzahlen in Rekonfigurationsraumvarianten für alle Betriebsmodi

	Vollständ. Graph		Minimierter Graph		Differenzen	
Rekonf.	Quelle	Ziel	Quelle	Ziel	Kosten	Qualität
N1.1	4701	*4519*	4701	*4470*	+14	+0,0291
N1.2	*4519*	4745	*4470*	4745	-2	-0,0291
N1.3	4745	4316	4745	4316	0	0,0000
N1.4	4316	4406	4316	4406	0	0,0000
	4701	**4406**	**4701**	**4406**	**+12**	**0,0000**
N7.1	4567	*4705*	4567	*4338*	+18	+0,0485
N7.2	*4705*	*4338*			+18	-0,0485
N7.3	4338	4401	4338	4401	0	0,0000
N7.4	4401	4381	4401	4381	0	0,0000
N7.5	4381	4362	4381	4362	0	0,0000
	4567	**4362**	**4567**	**4362**	**+36**	**0,0000**
N15.1	4726	*4745*	4726	*4450*	+10	+0,0337
N15.2			*4450*	*4401*	-6	-0,0070
N15.3	*4745*	4316	*4401*	4316	-4	-0,0267
N15.4	4316	4406	4316	4406	0	0,0000
	4726	**4406**	**4726**	**4406**	**0**	**0,0000**
					+16	**0,0000**

Über alle Betriebsmodi hinweg bleiben die Qualitäten stabil, die Rekonfigurationskosten steigen jedoch bei der Verwendung des vollständigen Graphens um durchschnittlich 16 Modifikationen. Die individuellen Werte zu den Alternativenvergleiche und Differenzen sind in der Tabelle B.6 in Anhang B abgelegt.

Aus den Ergebnissen lässt sich ableiten, dass auch bei einer umfangreichen Fehlermenge die qualitativ relevanten Konfigurationen nach der Minimierung erhalten bleiben. Im vollständigen *DARG* sind einzelne qualitativ hochwertigere Konfigurationen enthalten, die nicht auch im minimierten *DARG*

liegen. Die Alternativen sind aber kostenorientiert weit abgeschlagen und sollten daher nicht Teil des Suchraums sein. Das meta-heuristische Verfahren arbeitet somit effektiv und liefert einen relevanten Ausschnitt aus dem Entwurfsraum bei Vermeidung einer erschöpfenden Suche.

6.6.4.6. Metrik für Validierungsfrage FF4.3

FF4.3 *„Treten qualitative Abstufungen zwischen adäquaten Entwurfsalternativen in einem relevanten Maße in realen Systemen auf?“*

Die folgende Metrik überprüft das Vorliegen qualitativer Degradierungen zwischen ermittelten Alternativkonfigurationen in der Fallstudie. Die Degradierungen beeinflussen den Wartungsablauf signifikant, da sie eine Prognose über die qualitative Stabilität des Systems nach einer Rekonfiguration geben. Aus diesen Gründen müssen diese Informationen auch während der Abstraktion des realen Systems im Rahmen des vorgestellten Ansatzes erhalten bleiben.

Datencharakterisierung: Im variantenreichen Systementwurf liegen viele Konfigurationsalternativen vor, die sich durch einen unterschiedlichen Ressourceneinsatz in der erreichten Systemqualität unterscheiden. Mit Bezug auf das laufende System ist die Ressourcenplattform variabel und somit auch die Menge der einsetzbaren Konfigurationen. Dieser Raum wird zur Effizienzsteigerung im vorgestellten Ansatz minimiert und priorisiert. Dabei ist der Unterschied zwischen den Konfigurationen als qualitative Streuung messbar.

Datengrundlage: Die qualitative Bewertung der Konfigurationen aus Datenquelle 2 wird zur Ermittlung der qualitativen Streuung im Rekonfigurationsraum eingesetzt. Liegt eine hohe Streuung im Pfad vor, sind die Konfigurationen qualitativ sehr unterschiedlich. Findet während der Exploration des minimierten *DARG* eine qualitative Abwägung zwischen mehreren Alternativen innerhalb eines Schrittes im Rekonfigurationspfad statt, ist diese Nutzwertänderungen in Datenquelle 7 festgehalten. Die Datenquelle 1 wird verwendet, um Zusammenhänge zwischen Ressourcenanzahl und qualitativen Schwankungen zu untersuchen. Der Ansatz unterstützt die Identifikation

von Alternativen mit möglichst geringer Streuung, um eine entsprechende Stabilität zu gewährleisten. Um den Vorteil der qualitätsorientierten Festlegung auf eine Konfiguration zu argumentieren, wird auf diesen Aspekt zurückgegriffen.

Datenerhebung: Die Gesamtintensität der qualitativen Streuungen im Rekonfigurationsraum wird durch einen qualitativen Vergleich aller Konfigurationen im *DARG* für einen ausgewählten Betriebsmodus beurteilt. Diese graphbasierte Messung soll die generelle Existenz von Streuungen und deren Ausprägungen aufzeigen. Zusätzlich werden die Streuungen entlang der exemplarischen Fehlersequenz für alle verbundenen Alternativkonfigurationen pfadbasiert untersucht.

Ergebnis: Nachfolgend wird die qualitative Streuung examplarisch für den Rekonfigurationsraum von Betriebsmodus $N1$ untersucht. Die mehrdimensionale Messung der Streuungen der Qualitäten der Konfigurationen eines Rekonfigurationsraums zur Entwurfszeit zeigen die Dissimilaritäten entsprechend Abbildung 6.17 auf.

Die Daten werden in Form eines Dendrogramms dargestellt, welches zwischen allen Konfigurationen die euklidische Distanz im mehrdimensionalen Raum der Qualitätsattribute bestimmt und nach einer Clusteranalyse visualisiert.

Die y-Achse zeigt den Grad der qualitativen Abweichungen der Konfigurationsalternativen auf der x-Achse. Qualitativ besonders ähnliche Alternativen oder Gruppen von Alternativen bilden Cluster. Die Abweichung dieser Cluster zueinander wird im Diagramm durch vertikale Linien abgetragen. Insgesamt zeigt die Abbildung, dass ein hoher Grad von Streuungen (maximal rund 5,63 Dissimilarität) im gesamten Rekonfigurationsraum auftritt. Dies bedeutet zugleich, dass die Varianz im System bezüglich der Austauschbarkeit der Ressourcen ebenfalls hoch ist.

Im Kontext des konkreten Rekonfigurationpfads entlang der exemplarischen Fehlersequenz lässt sich für die hervorgehobenen Konfigurationen Folgendes beobachten.

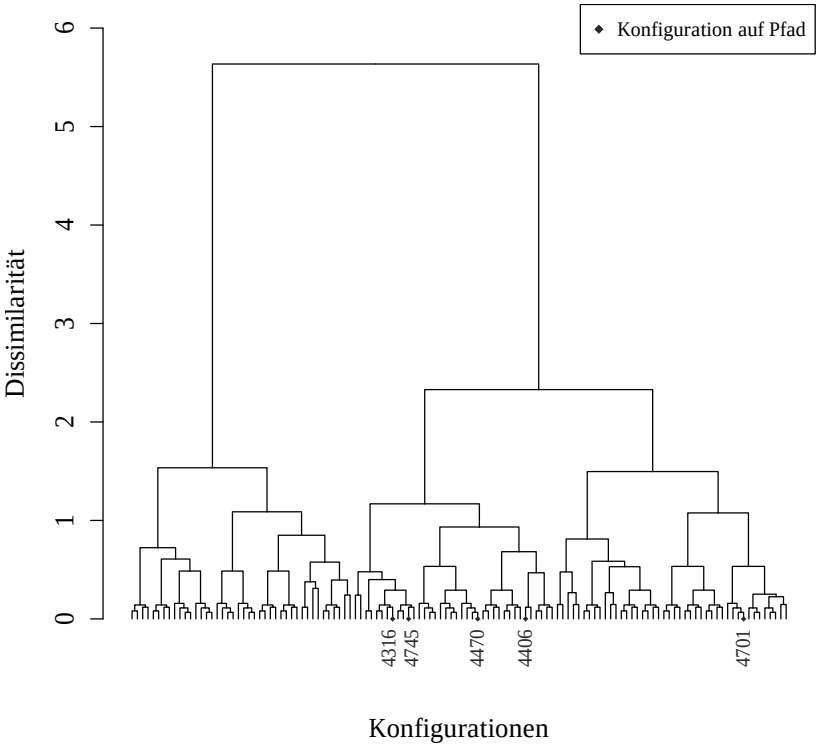


Abbildung 6.17.: Dissimilarität der Qualitätsbelegungen des Betriebsmodus N1

Die Konfigurationen im Pfad liegen mehrheitlich in einem Cluster mit einer Dissimilarität von circa 1,2. Die vom Anwender gewählte Startkonfiguration weicht davon ab und ist den Folgekonfigurationen zu circa 50% unähnlich (rund 2,5). Dieser initiale Wechsel in einen gänzlich anderen Cluster ist bei allen drei Betriebsmodi[26] messbar. Die initialen Konfigurationen wurde auf Grundlage der anfangs hochverfügbaren Ressourcenplattform ausgewählt. Die Messung zeigt jedoch, dass qualitativ ähnliche Architekturen zwar

[26] Die Dissimilaritäten für die Modi $N7$ und $N15$ sind in Abbildung B.1 beziehungsweise Abbildung B.2 in Anhang B dokumentiert.

vorliegen, ein Clusterwechsel auf Grundlage der Rekonfigurationskosten forciert wird. Weiter zeigen die Messdaten, dass die qualitative Priorisierung in der Meta-Heuristik die Systemqualität berücksichtigt und keine großen Clusterwechsel vorkommen.

Zum genaueren Vergleich der qualitativen Differenzen zwischen den Pfadkonfigurationen und deren Alternativen zeigt Abbildung 6.18 die Streuung für Modus $N1$[27]. Das Diagramm zeigt die qualitativen Degradierungen für die vier Rekonfigurationen für Betriebsmodus $N1$. Für jedes Segment im Rekonfigurationspfad (pro eingestreutem Fehler), wird die jeweilige qualitative Differenz zwischen Quell- und Zielkonfiguration hervorgehoben.

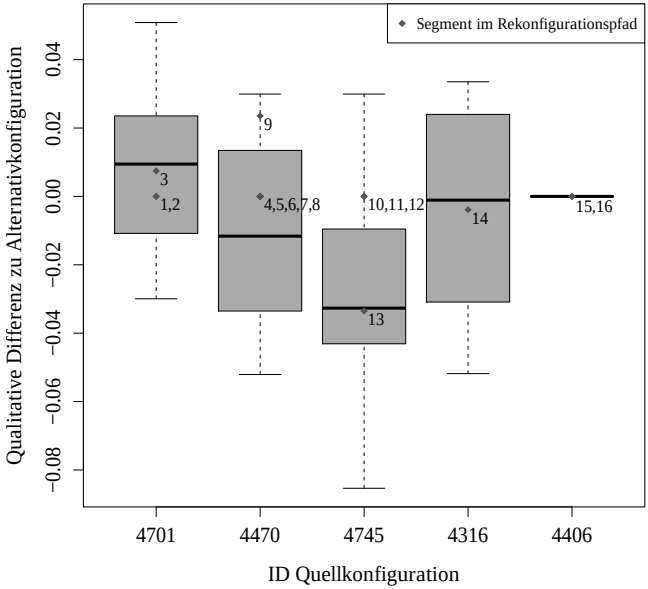


Abbildung 6.18.: Verteilung der Qualitätsdifferenzen des Betriebsmodus N1 entlang des Rekonfigurationspfads

[27] Abbildung B.3 und Abbildung B.4 in Anhang B zeigen die Streuungen für die Betriebsmodi $N7$ und $N15$.

Solange ein Fehler innerhalb einer Konfiguration kompensiert werden kann, werden keine Vergleiche durchgeführt und die Streuung beträgt 0,0, das heißt Quell- und Zielkonfiguration sind identisch. Sobald ein Konfigurationswechsel notwendig wird, tritt auch eine qualitative Differenz auf. Im Diagramm ist dies bei den Segmenten 3, 9, 3 und 14 (vergleiche Pfadlänge in Metrik 2.6) der Fall. Anschließend erfolgt die Rekonfiguration in die Zielarchitektur, die wiederum eine qualitativ abweichende Menge von Alternativen besitzt. Die letzte Konfiguration im Pfad (4406) bietet keine Streuungsinformationen, da keinerlei vergleiche durchgeführt werden bis zum Ende der Fehlersequenz. Das Diagramm zeigt, dass die Qualität in Einzelfällen durch eine Rekonfiguration gesteigert werden können. Dies ist damit zu begründen, dass zusätzliche Rekonfigurationskosten in Kauf genommen werden. Wird zusätzlich zur qualitativen Streuung der entsprechende Ressourceneinsatz hinzugezogen, ergibt sich die in Abbildung 6.19 gezeigte Darstellung des Rekonfigurationspfads[28].

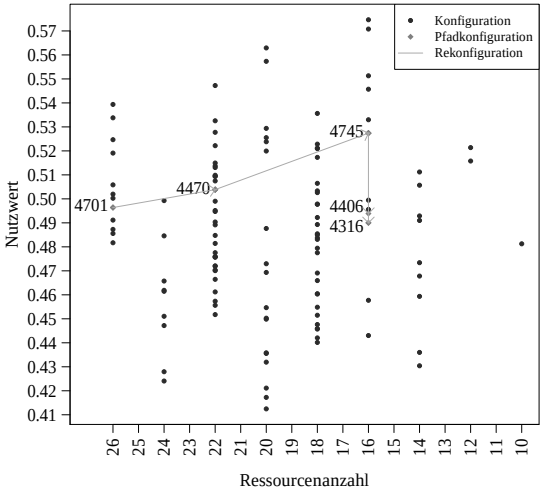


Abbildung 6.19.: Streuung der Gesamtqualitäten und Ressourcenmengen des Betriebsmodus N1

[28] Analog für $N7$ und $N15$ in Abbildung B.5 und Abbildung B.6 in Anhang B.

Das Streudiagramm zeigt die ermittelten Nutzwerte und Ressourcenmengen aller Konfigurationen im Rekonfigurationsraum. Dabei werden Ressourcenidentitäten und das Ausdünnen des Raums mit jedem weiteren Ressourcenausfall nicht betrachtet. Zusätzlich sind der Rekonfigurationspfad für die exemplarische Fehlersequenz von Betriebsmodus $N1$ hervorgehoben. Der Pfad beginnt mit Konfiguration 4701. Zunächst wird in den ersten beiden Konfigurationen der Nutzwert gesteigert. Zudem werden die Ressourcen um fast zehn Einheiten reduziert, um die qualitative Stabilität aufrecht zu halten. Beim Verlassen der Konfiguration 4745 sind die Rekonfigurationskosten allerdings zu hoch um in eine besser bewertete Konfigurationen zu wechseln. Der Pfad endet in Konfiguration 4406, in dem keine gültige Alternative ausgewählt werden kann. Die sonstigen Konfigurationen sind dann ressourcen-bedingt nicht mehr verfügbar.

Zur Vollständigkeit sind in Tabelle 6.11 die jeweiligen Änderungen der Ressourcenmengen mit jedem Änderungsschritt im betrachteten Pfad aufgeführt. Es werden nur die Differenzen aufgeführt die zum Zeitpunkt der Rekonfiguration vorliegen und den Wechsel auslösen.

Tabelle 6.11.: Ressourcendifferenzmengen der Pfadkonfigurationen des Betriebsmodus N1

Quelle	Ziel	Abzüge	Hinzunahmen
4701	4470	GPS Receiver2	∅
4470	4745	ASC DPU1 High Res.; CSS FrontHead1; CSS FrontHead2; CSS RearHead2; CSS Chipset2 Low Res.; CSS Chipset1 Low Res.; CSS RearHead1	∅
4745	4316	MC1x	∅
4316	4406	RW2	∅

Für den Betriebsmodus $N1$ treten nur Verringerungen der Ressourcenmengen auf, bei den Betriebsmodi $N7$ und $N15$ werden im Laufe des Reconfigurationspfads auch Ressourcen hinzugefügt, vergleiche Tabelle B.1 und Tabelle B.2 in Anhang B.

Unter der Annahme, dass eine hohe Variabilität in der Ressourcenplattform vorliegt, zeigt die Messung, dass in dem untersuchten Rekonfigurationsraum ein hohes Maß an Streuung auftritt. So lässt sich allgemein eine hohe Dissimilarität im minimierten *DARG* feststellen, wobei eine qualitative Degradierung für einen konkreten Rekonfigurationspfad feststellbar ist. Obwohl die Auswahl alternativer Konfigurationen durch Rekonfigurationskosten beschränkt wird, liegt eine hohe qualitative Stabilität in der Alternativenexploration vor.

6.7. Auswertung der Datenerhebung

Zur Beurteilung der Ergebnisse ist gemäß *GQM6*, vergleiche Abschnitt 6.1.2, eine Interpretation und Einordnung der qualitativen Aussagen in den Experteninterviews und den quantitativen Daten aus der werkzeuggestützten Messung notwendig. Dabei stehen jeweils die Adäquatheit der gesetzten Ziele und die Anwendbarkeit des Ansatzes im Vordergrund.

6.7.1. Auswertung der empirischen Studie

Die Interviews wurden erfolgreich durchgeführt. Jeweils konnte eine hohe Bereitschaft zur Freigabe von Informationen bestätigt werden. Weiterhin war die Deckung der Aussagen beider Experten hoch, wobei die Detailtiefe beidseitig variierte. Der Ansatz wurde grundsätzlich offen und positiv aufgenommen, was konstruktive Interviews unterstützte. Neben fachlichen Informationen wurden Einblicke in den allgemeinen Entwicklungsprozess von Raumflugkörpern gewährt. Dies beinhaltete vorrangig Details zu TET-1, sowie Vorgänger- und Nachfolgesystemen.

Auf Basis der Vorgaben aus Abschnitt 6.5.8 werden die erhobenen Aussagen aus den Interviews entsprechend ihrer Bedeutung zur Beantwortung der Validierungsfragen nachfolgend bewertet. Nach Kaiser [Kai14] werden Kernaussagen beider Experten interpretiert und mögliche Abweichungen zu den theoretischen Zielen des Ansatzes diskutiert.

6.7.1.1. Beurteilung der Validierungsfrage FF1.1

FF1.1 *„Ist eine Abstraktion des Laufzeitkontextes im realen Systementwurf legitim?“*

Eine Abstraktion des Laufzeitkontextes ist teilweise möglich, wobei eine Interaktion mit Experten zur Auswahl einer Lösungsalternative weiterhin notwendig bleibt. Die Experten ergänzen die Prognose des Systems mit dem Wissen aus ihrer Erfahrung.

> *„Ein Tool [für Entscheidungsträger], dass mentalitäts- und expertenunabhängig beratend zur Seite steht, ist vielleicht gar nicht so schlecht.“*
>
> — Systemingenieur(in)

Im aktuellen Prozess werden insbesondere in frühen Entwicklungsphasen bereits eine Vielzahl von Hardware- und Umgebungsmodellen genutzt, die jeweils von dem Ausführungskontext abstrahieren. In späteren Phasen sind Tests auf dem Prüfstand jedoch unumgänglich, um die Unschärfen in Prognosen zu minimieren.

> *„Was gemessen wird, ist die Wahrheit.“*
>
> — Systemingenieur(in)

Die Validierungsfrage ist somit *grundsätzlich positiv* zu beantworten, wobei Experten in Entscheidungen auf Basis von abstrakten Abschätzungen eingebunden werden sollten.

6.7.1.2. Beurteilung der Validierungsfrage FF1.2

FF1.2 *„Ist der Systemzustand nach Fehlereintritt eindeutig beschreibbar?“*

Die Abschätzung des Systemzustands nach Auftreten eines Fehlers ist für erwartete Fehlerfälle sehr gut möglich. Durch den Vergleich mehrerer gleichzeitiger Messungen und Modellwerte lassen sich Fehler detektieren und lokalisieren. Daraus wird der Zustand des Gesamtsystems abgeleitet

> *„Alle Informationen werden meistens nicht nur von einem Sensor abgearbeitet, sondern von mehreren. Dabei sind auch schon Entscheidungsmechanismen integriert.“*
>
> — Softwareingenieur(in)

Die Interpretation der Messdaten erfolgt durch die Experten und ist daher limitiert nach deren Verfügbarkeit und Erfahrungen.

> *„Die ganze Fehleranalyse ist sehr erfahrungsgetrieben."*
> — Softwareingenieur(in)

> *„Lessons learned gehen immer in so ein Projekt ein."*
> — Systemingenieur(in)

Insgesamt ist die Validierungsfrage *positiv* zu beantworten, da kontextbedingte Fehler erwartet werden und entsprechende Mechanismen zur deren Behandlung vorbereitet sind. Auch die Auswirkungen unerwarteter Fehler können durch mehrfache unabhängige Messungen potenziell identifiziert werden.

6.7.1.3. Beurteilung der Validierungsfrage FF1.3

FF1.3 *„In welchem Umfang werden Fehlerauswirkungsanalysen durchgeführt?"*

Der Umfang der Fehlerauswirkungsanalyse richtet sich nach dem Umfang und den Vorgaben der Ausfallsicherheit vom Kunden.

> *„Die Redundanzen werden vom Auftraggeber festgelegt. Wir können Empfehlungen abgeben, aber in der Regel sagen auch wir, dass alles redundant sein sollte."*
> — Systemingenieur(in)

Fehlerbaumanalysen dienen dabei als festes Instrument zu der Auswahl und der Kombination redundanter Hardwareressourcen. Dabei werden Informationen aus Datenblättern mit Erfahrungswerten kombiniert und den Missionsanforderungen gegenübergestellt. Zur Beurteilung des Betriebsverhaltens werden weiterhin Simulationsmodelle für Bauteilgruppen eingesetzt.

> *„Wir haben natürlich Simulationsmodelle (auf MATLAB/SimuLink-Basis) von den Rädern. Die haben aber nichts mit Reliability zu tun."*
> — Systemingenieur(in)

Im Laufe der Entwicklung verschiebt sich somit der Fokus jedoch auf Simulationen am Teststand. Fehlerauswirkungen werden somit zwar modellbasiert berechnet, aber hauptsächlich auf Hardwarebasis erprobt und beurteilt.

> *„Wenn möglich, nimmt man lieber Hardware. Es gibt aber auch viele Fehler, die man nicht mit Hardware abbilden kann, weil sich der Fehler in der Hardware nicht provozieren lässt."*
> — Softwareingenieur(in)

In Beantwortung der Validierungsfrage kann *ein umfangreicher Prozess zur Fehleranalyse festgestellt* werden.

6.7.1.4. Beurteilung der Validierungsfrage FF1.4

FF1.4 *„Ist eine statische Betrachtung von Qualitätseigenschaften ausreichend?"*

Eine ausschließliche statische Untersuchung von Qualitätseigenschaften aus Datenblättern findet im aktuellen Prozess bei der Grobauswahl der Ressourcen statt.

> *„Man sucht seine Komponenten danach aus, dass diese schon mal geflogen sind und man weiß [dann], bei dem Datenblatt steht was dahinter."*
> — Systemingenieur(in)

> *„Das [Verwendung von Datenblatt-Wissen] ist aber auch üblich in solchen Systemdesign-Studien."*
> — Softwareingenieur(in)

Umweltbedingungen werden dabei ebenfalls auf statische Werte, zum Beispiel Konstanten für Atmosphärenreibung und Gravitation, reduziert.

> *„Bei den üblichen Auslegungen und Berechnungen wird das [Umweltbedingungen] auch als konstant angesehen."*
> — Softwareingenieur(in)

Eine ausschließliche statische Untersuchung von Qualitätseigenschaften ist dennoch eher in frühen Entwurfsphasen, in Abhängigkeit der Verfügbarkeit von Prognosewissen, rentabel.

> *„Es kommt darauf an, wie gut das Input-Wissen ist [...] Das ist in diesen frühen Phasen hilfreich und in späteren natürlich auch, um Details sichtbar zu machen."*
> — Softwareingenieur(in)

In späteren Phasen sind qualitativ gestützte Vorschläge jedoch umfangreich zu begründen, um das Vertrauen für die Prognosen zu stärken.

> „*Wenn die Lösungen dicht an dem dran sind was jeder erwartet, dann wird das auch auf Akzeptanz stoßen.*“
> — Softwareingenieur(in)

> „*Solange das logisch argumentiert wird, ist das durchaus vorstellbar.*“
> — Systemingenieur(in)

Die Auswertung der Antworten auf die Validierungsfrage zeigen, dass *im aktuellen Prozess bereits statische Qualitätswerte* für Bauteile und Umweltbedingungen genutzt werden. Für späteren Entwicklungs- und Betriebsphasen wäre dies jedoch umfangreich zu argumentieren.

6.7.1.5. Beurteilung der Validierungsfrage FF2.1

FF2.1 „*Hängen personelle Aufwände primär von Diagnose- oder Reparaturtätigkeiten ab?*“

Die Aufwände von Diagnose und Reparatur hängen stark von der Lokalisierung und Reproduzierbarkeit des Fehlers ab. Eine wesentliche Einschränkung bei der Diagnose ist die Verfügbarkeit von Messpunkten im System.

> „*Im ungünstigen Fall fehlen genau die Werte, die benötigt werden, um den Fehler nachzustellen.*“
> — Softwareingenieur(in)

> „*Wir haben eine Menge Sensoren die unser System beobachten, aber es liegt [dennoch] nicht auf dem Tisch. Wir können immer nur das sehen, was wir gerade messen.*“
> — Systemingenieur(in)

Zur Unterstützung der Aufwände in der Reparatur eines Fehlers, wurde im TET-1 die Autonomie teilweise auch gesteigert.

> „*Wenn man beobachtet, dass ein Fehler öfter auftritt – Fehler muss dabei nicht gleich Komplettausfall bedeuten –, probiert man natürlich auch etwas mehr Autonomie zu integrieren, um die Bodenstation zu entlasten.*“
> — Systemingenieur(in)

Der Prozess bleibt jedoch sehr zeitaufwendig und hängt stark von der Verfügbarkeit und Qualifikation des Personals ab.

> *„Man braucht die Experten, die sich auskennen. Diese müssen es [den Fehler] dann auch fixen. […] Wenn zwischendurch der Experte nicht mehr verfügbar ist, hat man auch ein Problem. Es ist bei diesen Einzelentwicklungen [Spezialsystemen] immer so."*
> — Systemingenieur(in)

Abschließend bedarf jede Änderung im System auch einer Wiederholung einer Reihe von Diagnosen.

> *„Raumfahrt ist immer sehr testaufwendig. Das kostet meistens richtig [viel] Zeit."*
> — Systemingenieur(in)

Die Aufwände zwischen Diagnose und Reparatur sind für die Validierungsfrage *nicht eindeutig abgrenzbar*. In der Diagnose liegen Beschränkungen in der Datenverfügbarkeit vor. Jede Modifikation enthält in deren Erprobung auch wieder Diagnosephasen mit hohem Personaleinsatz.

6.7.1.6. Beurteilung der Validierungsfrage FF2.2

FF2.2 *„Resultiert der wesentliche Kommunikationsaufwand aus Datenmengen oder Latenzen im Systemzugriff?"*

Der Kommunikationsaufwand bezieht sich hauptsächlich auf den nicht durchgängigen Kontakt zwischen Satelliten und Bodensegment. Beim TET-1 sind beispielsweise nur zwei Kontakte pro Tag möglich, was die Diagnosezeit massiv beeinflusst.

> *„So kann es sein, dass ein Fehler erst nach 12 Stunden festgestellt wird."*
> — Softwareingenieur(in)

Die Übertragungskapazität ist zwar auch auf wenige Kilobits pro Sekunde beschränkt, fällt aber aufgrund der üblichen Wartungszeiten nicht ins Gewicht.

> *„Wenn man schon ein bis zwei Monate [für die Entwicklung] gebraucht hat, dann ist dieser eine Tag auch noch da."*
> — Softwareingenieur(in)

Im Idealfall ist eine Fehlerdiagnose und die Telekommandierung in einem Kontakt vorteilhaft.

Die Kommunikationsbeschränkungen liegen vorrangig in der Latenz, wobei vor allem die Lösungen innerhalb von einem Systemkontakt im Idealfall übertragen werden sollten. Somit ist die Validierungsfrage mit einem *hauptsächlich latenzbedingten Kommunikationsaufwand* zu beantworten.

6.7.1.7. Beurteilung der Validierungsfrage FF2.3

FF2.3 *„Lässt sich anhand vorliegender Diagnosedaten die verbleibende Konfigurierbarkeit des Systems nachvollziehen?“*

Die Diagnosedaten des TET-1 beschreiben den Zustand, die Aktivierung und den Betrieb aller Ressourcen. Die erweiterten Housekeeping-Daten liefern weitere Parameter zu den einzelnen Subsystemen. Dennoch ist eine Approximation des gesamten Systemzustands nur bedingt möglich.

> *„Es ist natürlich schwer nur anhand von Bits und Bytes [Hardware-]Fehler herauszukriegen, zu analysieren und einzusortieren.“*
> — Softwareingenieur(in)

Verbleibende Konfigurationen werden daraus nicht automatisiert erhoben, da nicht alle Subsysteme ständig aktiv sind. Stattdessen werden Situationen auf dem Teststand reproduziert, um den Raum möglicher Alternativen zu ermitteln.

> *„Es gibt keinen Satelliten, der perfekt und fertig ins All geschossen wird. Jeder weiß, dass es in der LEO-Phase[29] zu normalen Komplikationen kommt und dass man da irgendwas machen muss.“*
> — Systemingenieur(in)

Die verbleibende Konfigurierbarkeit des Systems ist den Experten grundsätzlich bekannt. Dieses Wissen ist allerdings nur implizit und bezieht sich nur auf die momentan aktivierten Baugruppen des Systems. Die Validierungsfrage ist daher nur *mit Einschränkungen positiv* zu beantworten.

6.7.1.8. Beurteilung der Validierungsfrage FF2.4

FF2.4 *„Ist ein modellbasierter Konfigurationsbegriff aus der Entwurfsphase für die Wartungsanalyse zur Laufzeit einsetzbar?“*

[29] Launch and Early Orbit Phase, initiale Betriebsphase nach dem Start des Flugsystems.

Die Entwicklung in der Domäne baut auf dem Wissen aus Vorgängermissionen auf. Entsprechend werden auch die Entwurfsmodelle mit jedem weiteren Projekt verfeinert.

> *„Man will nicht etwas Neues entwickeln, sondern man greift [wenn möglich] auf Altbewährtes zurück."*
> — Systemingenieur(in)

Bei Entwicklung von TET-1 wurden Erkenntnisse aus dem Betrieb eines technisch verwandten Vorgängers (BIRD[30]) in die Entwurfsmodelle integriert. Eine Verwendung und kontinuierliche Aktualisierung der Modelle zur Analyse eines Systems findet indes nicht statt.

> *„Wir prüfen erst [auf Alternativen], wenn der Anwendungsfall da ist. Wir untersuchen nicht vorher was wir eventuell noch alles [mit der Hardware] machen können."*
> — Softwareingenieur(in)

Die Betriebsphase ist somit entkoppelt von der Phase des Architekturentwurfs. Der Anwender müsste die Zusammenhänge von Quellcode und Architektur manuell herstellen. Die strukturierte Code-Basis des TET-1 fördert dabei jedoch strukturelle Bezüge herzustellen.

> *„Das [die Architektur] kann man ziemlich direkt aus dem Code ableiten."*
> — Systemingenieur(in)

Die Intensivierung des modellbasierten Konfigurationsbegriffs ist möglich und ein synchronisiertes Modell ist vielversprechend, um den Entwickler in der Auswahl von Wartungsaktionen zu unterstützen.

> *„Das denke ich, dass sowas sehr beliebt werden würde."*
> — Softwareingenieur(in)

Im aktuellen Wartungsprozess wird ein modellbasierter Konfigurationsbegriff nicht genutzt, wäre aber durch die architekturorientierte Quelltextstruktur gut integrierbar. Dies führt uns zu einer *positiven Beantwortung* der Validierungsfrage.

[30] Bi-spectral InfraRed Detection; Details zum Vorgänger von TET-1 im Earth Observation Portal: `https://directory.eoportal.org/web/eoportal/satellite-missions/b/bird`

6.7.1.9. Beurteilung der Validierungsfrage FF3.3

FF3.3 *„Ist der vorgestellte Ansatz in bestehende Wartungsprozesse zur Fehlerbehandlung integrierbar?“*

Eine Integration des Ansatzes ist möglich, wobei die Ausprägung als reines Beratungssystem zur Verkürzung der Reaktionszeiten deutlich bevorzugt wird.

> *„Dann stelle ich mir vor, dass die Entscheidungen schneller getroffen werden. […] und das vielleicht auch bessere Entscheidungen getroffen werden, weil […] die Qualität entscheidet und die Entscheidungsfinder eine objektive Meinung haben.“*
>
> — Systemingenieur(in)

> *„Aus Engineering-Sicht würde ich sagen, dass die meisten Ingenieure sich trotzdem über so etwas freuen würden. […] Für solche Systeme gut, weil es eben lange Reaktionszeiten gibt.“*
>
> — Softwareingenieur(in)

Der Entscheidungsraum sollte daher durch den Ansatz optimiert werden, jedoch müssen die Experten Einfluss auf die vorgeschlagenen Lösungen nehmen können.

> *„Wenn es scheitert, dann scheitert es daran, dass das System nicht genug Informationen hat. […] Es bringt mir nichts, wenn mein System [nach der Rekonfiguration] auf einem Stand von dem Design von vor zwei Jahren ist.“*
>
> — Systemingenieur(in)

> *„[…] unabhängig ob wir nun eine Wartung machen, gucken wir erstmal nach wie können wir überhaupt weiterarbeiten.“*
>
> — Softwareingenieur(in)

Dies beinhaltet auch eine qualitative Degradation von Missionszielen.

> *„Was bringt es einem wenn der Satellit in den Safemode geht bis es sich unten [in der Bodenstation] jemand anguckt und die Änderungen einpflegt und es gibt solange keine Daten. Dann hätte ich lieber ein autonomes System was mir irgendwelche Daten liefert in dem selben Zeitraum.“*
>
> — Systemingenieur(in)

Dennoch wären Anschlusstests vor einem Konfigurationswechsel unvermeidbar.

> *„Es muss klar sein, dass jede mögliche Konfiguration vorher getestet wurde."*
>
> — Systemingenieur(in)

Eine Realisierung für kommerzielle Missionen erscheint wegen kurzen Entwicklungszyklen unwahrscheinlich. In Forschungsmissionen müssen finanzielle Budgets eingehalten und das Vertrauen für Autonomie bestätigt werden.

> *„Der TET-1 durfte zu Anfang relativ wenig Autonomie haben und hat dann ziemlich viel dazubekommen."*
>
> — Systemingenieur(in)

Eine Steigerung des Entwicklungsaufwands in der Fehleranalyse wäre über die Steigerung der Zuverlässigkeit zu argumentieren.

> *„Wenn man das System dadurch sicherer macht, dann wird der Entwicklungsaufwand gerechtfertigt. Wenn man nur in der LEO-Phase dadurch Zeit spart, dann ist das Verhältnis wichtig."*
>
> — Systemingenieur(in)

Der Einsatz des Ansatzes muss den Stand der Technik voranbringen und eine umfassende Exploration beinhalten.

> *„Ich denke aber auch, dass man das teilweise auch durch normale Engineering-Arbeit abdeckt, da immer mehrere Fehlerlevel betrachtet werden. Wobei hier auch nicht alle anderen Wege berücksichtigt werden."*
>
> — Softwareingenieur(in)

Die Auswertung zeigt, dass eine Integration in die Fehlerbehandlung möglich ist, dies jedoch aus Gründen der Autonomie nur als Beratungssystem realistisch vertretbar ist. Die Validierungsfrage ist somit *diesbezüglich positiv* zu beantworten.

6.7.1.10. Beurteilung der Validierungsfrage FF3.4

FF3.4 *„Ist eine vorrangig modellbasierte Dokumentation der Wartungsschritte zielführend?"*

Die Dokumentation der Aktivitäten im aktuellen Wartungsprozess erfolgt textbasiert, wobei Wertabweichungen in den Housekeeping-Daten erfasst werden. Die Auswertung der Daten findet verteilt bei den zuständigen Entwicklern der Bauteilgruppen statt.

> *„Was unsere Produkte angeht wird das auf jeden Fall getrackt. […] Es liegt [dann] bei den Entwicklern, die Informationen an die Satellitenbetreiber weiterzuleiten.“*
> — Systemingenieur(in)

Eine modellbasierte und zentrale Dokumentation wäre zielführend, um den Gesamtüberblick für alle Beteiligten in diesem Prozess zu steigern.

> *„Eine Grafik wäre hilfreich – wo komme ich her, wo komme ich hin – weil man es [die Alternativen und Konsequenzen] schneller sieht.“*
> — Softwareingenieur(in)

Dazu sind die Informationen jedoch zunächst für alle Bauteilgruppen zusammenzuführen und zu abstrahieren, um unterschiedliche Fehlerausprägungen abzubilden. Dazu müssen Änderungen zunächst auf struktureller Ebene sichtbar gemacht werden. Generell sollte das System einen großen Raum an Möglichkeiten bieten, um die Vorteile des Ansatzes zu nutzen.

> *„Wenn ich nur fünf mögliche Konfigurationen habe, dann kann es ja auch sein, dass das am Boden vordefiniert ist. Wenn erst [Konfiguration] 5 gewählt ist und später [Konfiguration] 4, dann brauche ich nicht mehr Informationen dazu.“*
> — Softwareingenieur(in)

Die Frage zur Notwendigkeit einer modellbasierten Dokumentation der Wartungsschritte ist positiv zu beantworten, wobei der zusätzliche Aufwand nur gerechtfertigt werden kann, wenn das System variantenreich ist.

6.7.1.11. Beurteilung der Validierungsfrage FF4.2

FF4.2 *„Lassen sich Differenzen von Konfigurationen auf rein struktureller und qualitativer Ebene adäquat beurteilen?“*

In der vorliegenden Implementierung des ACS liegen qualitative Abstufungen auf Ebene der Hardware und in Softwaremodellen vor.

> *„[Die Sternkamera oder der Sonnensensor mit Magnetfeldsensor] kann die selbe Ausrichtung einnehmen, allerdings mit unterschiedlicher Qualität.“*
>
> — Softwareingenieur(in)

Weiterhin überbrücken modellbasierte Berechnungen temporäre Ausfälle. Um ein übermäßiges Abdriften der tatsächlichen Werte zu vermeiden, sind die Zeiträume gering. Beide Methoden werden nur als Rückfallebene genutzt, wenn das primäre Subsystem ausfällt. Grundsätzlich ist es notwendig eine Ressource redundant zu ersetzen, um die Missionsziele zu erreichen.

> *„Wenn die Komponente korrumpiert ist, dann hat diese die gleiche Qualität wie die [alternative Komponente].“*
>
> — Systemingenieur(in)

Im Normalfall würde die Qualität und auch die Struktur des Systems identisch bleiben. Es wäre keine Rekonfiguration beobachtbar.

Wird für einen Benutzungsmodus (Mission) die flexible Auswahl von Ressourcenkombinationen zugelassen, ist ein produktiver Betrieb mit unterschiedlichen Qualitäten und Ressourcen denkbar, sofern der Anwender darüber informiert wird. Mit der qualitativen Abweichung wären dann auch strukturelle Unterschiede beobachtbar.

> *„Das andere ist, dass natürlich dem Benutzer am Boden sichtbar gemacht wird [...] welche Konfiguration gerade gewählt wurde [...] [und] welche Qualität eigentlich gerade erreicht ist. Dann kann er entscheiden, ob er die Mission so fortsetzen möchte oder ob die Qualität nicht mehr ausreichend ist.“*
>
> — Softwareingenieur(in)

Die Antworten zu dieser Validierungsfrage zeigen auf, dass im Normalfall von Wechseln zwischen homogenen Ressourcen keine Differenz von Konfigurationen festgestellt werden kann. Werden qualitative Abstufungen allerdings zugelassen, werden Änderungen sichtbar. Entsprechend kann die Frage *grundsätzlich bestätigt* werden, solange eine Ausweitung der Missionsanforderungen angenommen wird.

6.7.2. Auswertung der werkzeuggestützten Messungen

Die Ergebnisse der Validierung werden auf Grundlage der werkzeuggestützten Messergebnisse aus Abschnitt 6.6.4 nachfolgend beurteilt. Im Vergleich zu dem aktuellen Entwicklungs- und Wartungsprozess wird die Adäquatheit des Ansatzes überprüft.

6.7.2.1. Beurteilung der Validierungsfrage FF2.5

FF2.5 *„Ist der Kommunikationsaufwand im Betrieb durch frühzeitige Analysen des Entwurfsraums reduzierbar?“*

Im Rahmen dieser Validierungsfrage wurde der theoretisch maximale Kommunikationsaufwand in der Wartung, bei Verwendung des *DARG*-Ansatzes in Abschnitt 6.6.4.1, approximativ erhoben. Zur Beurteilung einer signifikanten Aufwandsreduzierung, bezüglich der Kommunikation zwischen Satellit und Bodensegment ist die Messung mit dem aktuellen Vorgehen in der Wartung zu vergleichen. Der idealtypische Ablauf von der Meldung bis zur Behebung eines Fehlers ist im Sequenzdiagramm in Abbildung 6.20 exemplarisch dargestellt.

Analog zur Messung der Metrik wird hier der Ausfall von ausschließlich zwei Ressourcen untersucht. Konkret wird angenommen, dass in dem Reaktionsradsystem bereits vormals das einzelne Reaktionsrad RW3 ausgefallen ist und dies systemintern durch Redundanzen RW4 ausgeglichen wurde. Einen weiteren Ausfall kann das ACS nicht autonom kompensieren und der Safe-Modus wird eingeleitet. Konkret zeigt das Diagramm die Situation beim Ausfall des zweiten Reaktionsrades.

Laut dem Anforderungsdokument des ACS[31], werden bei der Kommunikation zwischen Satelliten und Bodensegment zunächst in jedem Sendezyklus alle Zustände über Gesundheit und Aktivierung der Ressourcen als Teil der Housekeeping-Daten übertragen. Dies führt zu einem Datensatz über die Zustände von bis zu 32 *Ressourcen*. Unter der Annahme einer 6-Bit-Kodierung der Ressourcen-ID analog zur Messung, führt dies zu $32 \times 6b = 192b$ *an Daten* und weiterhin zu jeweils einem Bit pro Zustand.

[31] Ausschnitte aus dem Originaldokument, untersucht von Märtin et al. [MSH+13]

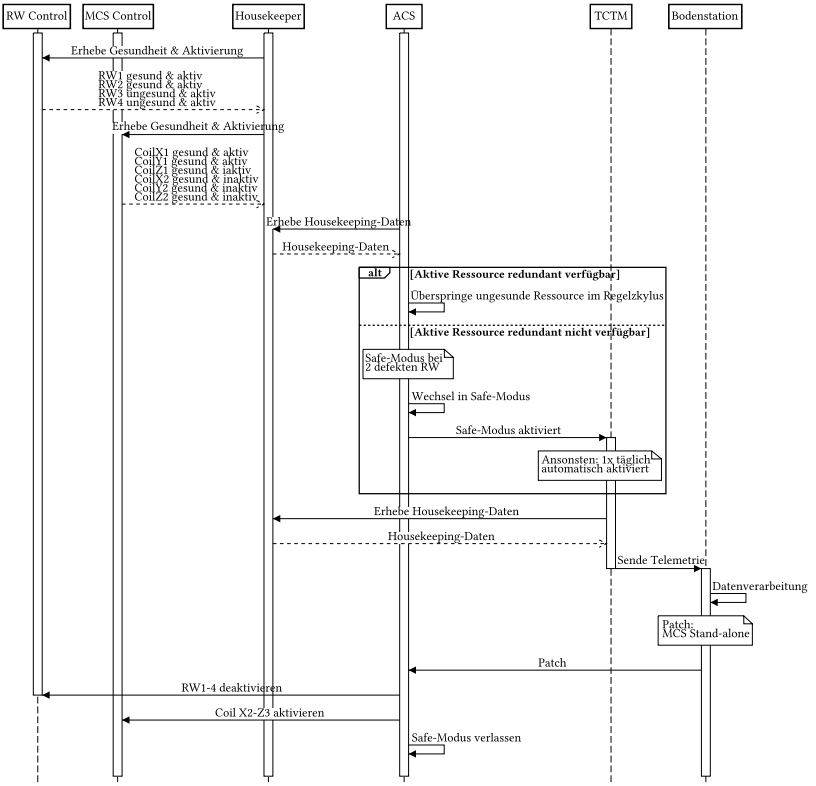


Abbildung 6.20.: Exemplarischer Ablauf einer Wartungsaktivität des TET-1 ACS

Nach der Durchführung der Fehleranalyse und Identifikation einer Rekonfigurationssalternative, werden die Änderungen (Englisch: Patch) vom Bodensegment auf den Satelliten übertragen. Im Beispiel wird der Betrieb komplett auf das Magnetspulensystem verlagert. Als Datenmenge sei dabei von ausschließlich notwendigen Änderungen in der Ressourcenplattform ausgegangen: $4 \times 6b = 24b$ zur Abschaltung der Reaktionsräder und $3 \times 6b = 18b$ zur Aktivierung zusätzlicher drei Magnetspulen. Die Pfeilrichtung zeigt die Kommunikationsrichtung an; vom beziehungsweise zum Satelliten.

	192	b	→ IDs aller Ressource
+	32	b	→ Gesundheitszustände
+	32	b	→ Aktivierungszustände
+	24	b	← IDs abgeschalteter Ressourcen
+	18	b	← IDs zugeschalteter Ressourcen
	298	b	Gesamte Datenmenge

Durch die ständige Übertragung aller Zustände entsteht ein Kommunikationsaufwand von *mehr als* 37 *Bytes* pro Meldung. Dabei wird von der Datenmenge zur Softwareparametrisierung abstrahiert.

Im Vergleich zur Nutzung des *DARG*-Ansatzes resultiert ein um *Faktor 12* (3:37 Bytes) höherer Kommunikationsaufwand. Werden umfangreichere Konfigurationen durchgeführt und die Änderungen in der funktionalen Architektur zusätzlich berücksichtigt, steigt der Faktor entsprechend. Durch das Vorhalten eines zum Produktivsystem synchronisierten Modells über Systemzustand und Rekonfigurierbarkeit, kann der Aufwand in der Datenübermittlung signifikant gesenkt werden.

Durch die Analyse im Vorfeld ist keine kontinuierliche Übertragung aller Housekeeping-Daten zu jeder Ressource notwendig, da diese Informationen in der Modellinstanz des *DARG* persistent verfügbar sind. Die hohen Kosten zur Erstellung des Modells zur Entwurfszeit und Analysen zur Vorausberechnung möglicher Alternativen, werden so während der Laufzeitwartung mit jedem auftretenden Fehler zunehmend kompensiert.

6.7.2.2. Beurteilung der Validierungsfrage FF2.6

FF2.6 *„Führt eine Priorisierung von Rekonfigurationsentscheidungen zu einer Reduktion des personellen Wartungsaufwands?“*

Der Wartungsaufwand unter Verwendung des *DARG*-Ansatzes wird in Abschnitt 6.6.4.2 für alle Betriebsmodi ermittelt. Die Abwägung und Priorisierung von alternativen Lösungen hin zu einem eindeutigen Lösungskandidaten wird dabei vollständig automatisiert. Ob dies auch zu einer signifikanten Verringerung der manuellen Aktivitäten in der realen Wartung führt, wird nachfolgend beurteilt. Es wird hier kein Bezug auf einen spezifischen Betriebsmodus genommen, sondern die Durchschnittswerte aus der Messung genutzt.

Das jederzeit verfügbare Wissen über den effektiven Raum möglicher Rekonfigurationen ist nach heutigem Stand der Technik nicht im Software-Wartungsprozess für Raumflugkörper, vgl. *ECSS Space Engineering: Software Engineering Handbook* [Eur13], integriert. Auch wenn der Entwicklungsstandard *ECSS-E-ST-40C* [Eur09] das Aufgreifen von Methodiken und Dokumentationsartefakten aus der Entwicklungsphase in der Softwarewartung beschreibt, ist der Entwicklungsprozess auf die traditionelle Einzelsystementwicklung mit starken Hardwarebezug ausgerichtet. Die Entwicklung erfüllt die technischen Anforderungen nach mehrfacher Fehlertoleranz durch den Einbau redundanter Bauteile. Hieraus wird zugleich die Rekonfigurationsfähigkeit des Systems abgeleitet. Nach unserem besten Wissen werden in der Entwurfsphase keine explorativen Analysen für *jegliche* ausführbare Abwandlungen des Systems unternommen. Somit wird der Konfigurationsraum nur implizit beschrieben und es liegen keine weitreichenden Informationen über mögliche Konfigurationsalternativen vor. Tritt im Betrieb ein Fehler auf, wird eine Alternative nur innerhalb der bekannten redundanten Ressourcenkonstellationen ermittelt. Traditionell werden Lösungskandidaten in Simulationen und auf dem Prüfstand (Englisch: Engineering Model) des Systems in Messungen gegeneinander abgewägt. Findet sich hier keine Lösung auf Ressourcenebene, verbleibt die erschöpfende Suche nach einer softwarebasierten Alternative im gesamten Entwurfsraum. Hier fehlt es allerdings an einer qualitativen Steuerung des Suchprozesses.

Ein allgemeiner Vergleich des Ansatzes mit dem existierenden Wartungsprozess ist durch die inhärente Heterogenität beider Prozesse nur begrenzt möglich. Während der existierende Wartungsprozess starke Bezüge zum Laufzeitverhalten der Ressourcen hat, stützt sich der *DARG*-Ansatz auf Prognosen der Systemqualität unter dem Einsatz von Datenblattwissen der Ressourcen. Eine Simulation oder ein Teststandversuch sind personell aufwendig, eine Prognose aus der Entwurfszeit unterliegt hingegen einer gewissen Unschärfe. Aus der *DARG*-Analyse der drei Betriebsmodi folgt, dass durchschnittliche insgesamt 4 Rekonfigurationen als Reaktion auf die exemplarische Fehlersequenz durchzuführen sind. Durch die absteigende Sortierung der Alternativen kann das Personal im besten Fall nur noch die jeweils erste Konfigurationsalternative einsetzen. Unter der Annahme, dass die Vorauswahl möglicher Alternativen zwar akzeptiert wird, die Alternativen jedoch bezüglich ihres Laufzeitkontextes im Prüfstand inspiziert werden, entstehen die folgenden Wartungsaufwände. Es wird weiterhin von den

durchschnittlich rund 23 gemessenen Änderungen in der Ressourcenplattform ausgegangen. Ohne Akzeptanz des Ansatzes sind für 79 erreichbare Konfigurationen

$$\frac{23\ \textit{Ressourcenänderungen}}{4\ \textit{Rekonfigurationen}} + \frac{79\ \textit{Konfigurationsvergleiche}}{4\ \textit{Rekonfigurationen}} = 26$$

Wartungsschritte als Annäherung an Status Quo notwendig. Wird hingegen der Ansatz gering akzeptiert, kann die Effektivität des Prozesses auf

$$\frac{23\ \textit{Ressourcenänderungen}}{4\ \textit{Rekonfigurationen}} + \frac{16\ \textit{Konfigurationsvergleiche}}{4\ \textit{Rekonfigurationen}} = 10$$

Wartungsschritte gesteigert werden, da nur noch strukturell naheliegende Alternativen überprüft werden. Bei hoher Akzeptanz des Ansatzes kann neben der Effektivität auch die Effizienz gesteigert werden, da in

$$\frac{23\ \textit{Ressourcenänderungen}}{4\ \textit{Rekonfigurationen}} + \frac{10\ \textit{Konfigurationsvergleiche}}{4\ \textit{Rekonfigurationen}} = 8$$

Wartungsschritte ausschließlich qualitativ relevante Alternativen verglichen werden.

Insbesondere bei der präzisen Abwägung von minimalen Ressourcenänderungen, ist der Einsatz von Simulationen und des Prüfstands in der Abwägung von Konfigurationsalternativen unabdingbar. Der *DARG*-Ansatz ist dann einsetzbar, wenn eine hohe Varianz in der eingesetzten Ressourcenplattform existiert. Gilt diese Annahme, sind zugleich eine hohe Anzahl möglicher Parametrierungen in der Erprobung möglich und somit die Überprüfung des Rekonfigurationsraum personell aufwendig. Im Vergleich ergibt sich vor allem durch die Betrachtung der Ressourcendifferenzen und den Nutzwerten der Konfigurationen ein Vorteil. Der *DARG*-Ansatz automatisiert nicht den gesamten Wartungsprozess, sondern setzt bei der Verkleinerung und Priorisierung des Entscheidungsraums für Konfigurationsalternativen an. So können Parametrierungen und Testszenarien gezielt vorausgewählt und anschließend mit den Laufzeitdaten manuell erprobt werden. Weiterhin erlaubt der Ansatz die Minimierung der Menge der Rekonfigurationen dadurch, dass die Suche in einem definierten Kostenrahmen für Ressourcenänderungen erfolgt. Neben zeitlichen Ersparnissen sind so auch Einsparung des Wartungspersonals möglich. Eine nähere Quantisierung der eingesparten Wartungsaufwände ist nicht möglich, da keine belastbaren Vergleichsdaten aus der Fallstudie vorliegen.

6.7.2.3. Beurteilung der Validierungsfrage FF3.1

FF3.1 *„Ist eine kompakte visuelle Darstellung des Rekonfigurationsraums umsetzbar?“*

In Abschnitt Abschnitt 6.6.4.3 wird eine kompakte Repräsentation des Rekonfigurationsraums als *DARG* anhand unterschiedlicher Graphkennzahlen beziffert. Im Vergleich werden nachfolgend Form und Ausprägung der vorliegenden Daten im Wartungsprozess näher untersucht.

Im TET-1 erhebt das Unterstützungssystem für die Nutzlast (Englisch: Payload Support System, PSS) alle relevanten Systeminformationen und protokolliert diese lokal. Dabei werden unter anderem Messdaten über den Verlauf der Betriebsmodi (Experimente) sowie die Housekeeping-Daten, vergleiche Abschnitt 6.4.6, erhoben und temporär festgehalten[32]. Aus dem Quellcode des Systems[33] geht bevor, dass diese Diagnosedaten als Vektoren abgelegt und als Telemetrie zur Bodenstation übertragen werden. Diese technische Sicht auf das System dient zur Beobachtung aktueller Gesundheits- und Betriebszustände zur Systemlaufzeit. In Vergleichen von Soll- und Istwerten dienen Abweichungen außerhalb gültiger Toleranzen als Indikatoren für Fehlerfälle. Die Indikatoren unterstützten das Personal in der Identifikation der Fehlerursachen und der Eingrenzung von Fehlerauswirkungen.

Der aktuelle Wartungsprozess sieht keine Phase zur Reevaluierung gültiger Konfigurationsalternativen auf Basis der Entwurfsmodelle vor. Die Diagnosedaten liefern detaillierte Informationen über die Verfügbarkeit konkreter Ressourcen sowie deren Verwendung in der aktuellen Systemkonfiguration. Im Falle eines Wegfalls einer redundanten Ressource, ist die Identifikation der alternativen Ressourcen konstruktiv vorgegeben. Dies gilt solange, bis die Redundanzen erschöpft sind, anschließend ist eine weitreichende Suche nach anderen Lösungen, das heißt Konfigurationsalternativen, notwendig. Ein *DARG* adressiert dazu vorrangig die Relationen zwischen Variationen von Software- und Ressourcenkonstellationen, nutzt dabei jedoch symbolische Ressourcenverweise. So verweist ein Knoten auf mehrere Gruppen redundanter Ressourcenoptionen, vergleiche Abschnitt 4.1.2. Die Diagnosedaten beschränken indes den Rekonfigurationsraum durch die *Konkretisierung ungesunder Ressourcen*. Der *DARG* leistet anschließend die Verknüpfung des

[32] `https://directory.eoportal.org/web/eoportal/satellite-missions/t/tet-1`
[33] Gekürzte Implementierung TET-1 ACS, untersucht von Märtin et al. [MSH+13]

Entwicklungswissens über den Rekonfigurationsraum und dessen Reduktion entlang des neuen Systemzustands zur Laufzeit.

Der *DARG* synchronisiert sich entlang der Ressourceninformationen aus den Diagnosedaten und liefert pro Systemzustand und gewähltem Betriebsmodus eine priorisierte Auswahl möglicher Konfigurationen, sofern möglich. Die Messergebnisse in Abschnitt 6.6.4.3 deuten darauf hin, dass durch die Einschränkung auf einen Teil der Ressourcenplattform und eines konkreten Betriebsmodus die Größe des *DARG* in einem kompakten Rahmen gehalten werden kann. Das Vorhalten eines kontinuierlich abgeglichenen Modelles des Rekonfigurationsraums unterstützt den vorliegenden Wartungsprozess mit signifikanten Kennzahlen. Dabei wird eine kompakte Präsentation durch Minimierungen der Kanten erreicht.

6.7.2.4. Beurteilung der Validierungsfrage FF3.2

FF3.2 *„Sind Zusammenhänge zwischen Fehlereintritt und Fehlerauswirkung nachvollziehbar darstellbar?“*

In der Untersuchung von Rekonfigurationen, werden in Abschnitt 6.6.4.4 vorrangig Kennzahlen für Pfade durch die *DARG*-Instanz quantitativ erhoben. Diese Daten geben einen Aufschluss darüber in welcher strukturellen Nähe sich gültige Alternativentwürfe befinden. Dies wird im Wartungsprozess insbesondere dann relevant, sobald die Redundanzoptionen einer Ressourcenkonstellation vollständig ausfallen. Die Herausforderung besteht dann darin eine potenziell komplexe Konfigurationsänderung für den Anwender nachvollziehbar darzulegen.

In den Diagnosedaten aus der Telemetrie des TET-1 sind Assoziationen nicht explizit adressiert. Die Zusammenhänge zwischen Konfigurationen sind dem Wartungspersonal als Domänenexperten zwar bekannt, eine explizite Manifestation dieses Wissens in einem Laufzeitmodell findet jedoch nicht statt. Die vorherigen Metriken haben gezeigt, dass die Komplexität des Variantenraums hoch ist und folglich bei manueller Exploration die kognitive Belastungsgrenze bereits für kleine Systeme schnell erreicht ist. Der inhärente Zeitdruck im Wartungsprozess, zum Beispiel bei notwendigen Erprobungen auf dem Teststand, lassen zusätzlich kaum Raum für ein Abwägen aller Konfigurationsalternativen.

Unter der Annahme, dass das System unter der Maßgabe der höchstmöglichen Fehlertoleranz im gegebenen Kostenrahmen entworfen wird, erfolgen im Entwurf aufwendige Untersuchungen zur Identifikation besonders belasteter Elemente, vergleiche [Eur13]. Neben der Duplizierung von Ressourcen, werden grundsätzlich weitere Rückfallebenen betrachtet, die einen Betrieb mit verminderter Leistung zulassen. So wurde auch in den Anforderungen des TET-1 ACS neben der *Präzisionsregelung* mit Reaktionsrädern und Magnetspulen, zusätzlich eine *grobe Regelung*, ausschließlich mit den Magnetspulen, beschrieben. Durch den Austausch zwischen beziehungsweise der Überschneidung von Entwicklungs- und Wartungspersonal ist dieses Wissen aus der Anforderungsphase grundsätzlich auch im Betrieb verfügbar. Allerdings sind diese Informationen in Dokumenten festgehalten und nicht innerhalb ausführbarer Analysemodelle. Der *DARG* bietet eine solche graphische und analysierbare Datenbasis, die sich entsprechend der Laufzeitdaten an den Systemzustand anpasst. Der Ansatz liefert dabei eine qualitätsorientierte Argumentation für die Zu- oder Abwahl diverser Ressourcen entlang einer exemplarischen Fehlersequenz und deren teilweisen Permutation.

Die Rekonfigurationspfade im *DARG* lassen sich für strukturell ähnliche Konfigurationen durch geringe Schrittweiten und Pfadlängen für den Domänenexperten entsprechend Abschnitt 6.6.4.4 nachvollziehbar darstellen. Der Ansatz begründet die An- oder Abwahl diverser Ressourcen durch die strukturelle Nähe und die angestrebte qualitative Stabilität. Naheliegende Assoziationen sind dem erfahrenen Personal dabei erwartungsgemäß bekannt. Kommt es allerdings zur Notwendigkeit einer umfassenden Änderung der Software- und Hardwarekonstellation, ist die Anwendung des Ansatzes naheliegend. Die Untersuchung der optimalen und kritischen Fehlersequenzen zeigt, dass der Ansatz auch bei Variationen der Fehlerreihenfolge gültige und erklärbare Ergebnisse liefert. Sind die Streuungen der Rekonfigurationskosten im gesamten Pfad sehr hoch, ist eine vorgeschlagene Konfigurationswahl plausibel zu begründen. Ähneln sich die Daten dabei, lässt die Unschärfe der Qualitätsprognose allerdings Zweifel am Analyseergebnis der Graphexploration.

6.7.2.5. Beurteilung der Validierungsfrage FF4.1

FF4.1 *„Ermittelt ein metaheuristisches Suchverfahren einen relevanten Ausschnitt des alternativen Entwurfsraums?“*

In der Metrik im Abschnitt 6.6.4.5 wird die Ergebnisqualität der Alternativenexploration im minimierten *DARG* mit der erschöpfenden Suche im vollständigen *DARG* verglichen. Nachfolgend soll beurteilt werden, ob diese rechenintensive Messung mit einem realen Wartungsprozess vergleichbar ist.

Wie zuvor festgehalten, ist der Grad der Expertise des Wartungspersonals in einer spezialisierten Domäne im Allgemeinen hoch und das Wissen über die möglichen Variationen im System sehr ausgeprägt. Zuvor wurde herausgearbeitet, dass speziell das Auffinden der Assoziationen eine Herausforderung darstellt; hier als Kanten im *DARG* manifestiert. In dieser Beurteilung stehen vorrangig die möglichen Konfigurationen in Abhängigkeit zur verfügbaren Ressourcenplattform, als Knoten im *DARG*, im Fokus. In den Unterlagen zur Fallstudie liegen keine konkreten Zahlen über die bekannten Konfigurationen vor, deshalb werden die Vergleichsdaten aus Annahmen zum Entwicklungsprozess extrapoliert.

Der vorliegende Vergleich bezieht sich nicht auf Information aus dem tatsächlichen Wartungsprozess, sondern orientiert sich an der Obergrenze für den gültigen Rekonfigurationsraum. Der *DARG* bezieht sich in beiden Ausprägungen bereits nur auf einen Teil des möglichen Entwurfsraums. Durch den Einsatz evolutionärer Algorithmen entfallen so etliche nicht-optimalen Kandidaten in der Alternativenbewertung im Voraus, vergleiche Abschnitt 3. Eine zusätzliche Raumverkleinerung erfolgt im minimierten *DARG* durch die Vorgabe der maximalen Änderungsschritte von einer Konfiguration zur nächsten. Der minimierte Suchraum priorisiert dabei grundsätzlich qualitativ vorteilhafte Konfigurationen und optimiert die Exploration hinsichtlich minimaler Rekonfigurationskosten. Diese beiden Minimierungsziele finden sich auch in üblichen Entwicklungsprozessen wieder um technische Schulden,vergleiche [Lil16], gering zu halten. Als Ausprägung der Qualitätssicherung validieren Tests oder Simulationen dabei den Erfolg einer Konfiguration. Insbesondere im Bezug auf die Parametrisierung eines Teststands, spielen die Rekonfigurationskosten bei Software-/Hardwaresystemen eine große Rolle. Im bestmöglichen Fall kann das Wartungspersonal alle vorteilhaften Konfigurationen vorbestimmen. Zur Eingrenzung der Auswahl würden jedoch auch qualitative Prognosen herangezogen werden. Entsprechend sei hier weiter angenommen, dass alle vorteilhaften Konfigurationen zur Entwurfszeit vollständig im „Sampling“selektiert wurden. Eine erschöpfende Suche in diesem gesamten Rekonfigurationsraum entspricht somit approxi-

mativ der Alternativensuche des Wartungspersonals. Die Ergebnismenge wird allerdings über die Rekonfigurationskostengrenze eingegrenzt, da diese in jedem Fall im kostenorientierten Wartungsprozess gilt. Dies harmoniert mit dem Minimierungsbegriff für einen *DARG*.

Der direkte Vergleich zum echten Wartungsprozess ist für die Fallstudie nicht möglich. Die Annahme einer erschöpfenden Suche ist bei größeren Systemen algorithmisch in annehmbarer Zeit nicht haltbar. Folglich ist der vorliegende Validierungsaspekt nur im Fallstudienkontext zu beantworten. Von zufälligen Kombinationen in der Suche (Englisch: Random Search) wird hier abgesehen, weil dies nicht einer der Sichtweise eines Domänenexperten, der auf das betrachtete System spezialisiert ist, entspricht. Die Ergebnisse der Metrik zeigen auf, dass auch eine frühe Restriktion des exemplarischen Rekonfigurationsraums zu keinen eklatanten Einschnitten in der Rekonfigurationsfreiheit führt, obwohl bis zu 90% der Kanten zu Alternativen minimiert werden. Dies hängt unmittelbar damit zusammen, dass in der vorliegenden Fallstudie bereits zur Entwurfszeit Wissen über die qualitativen Einflüsse einzelner Ressourcenverfügbarkeiten existiert. Dies ist in der Raumfahrtdomäne insbesondere dadurch möglich, da die eingesetzten Ressourcen üblicherweise bereits im Orbit geprobt (In-Orbit-Verifikation) wurden und somit Messdaten aus Vorläufersystemen vorliegen. In dieser Arbeit wird zur Generalisierung und Abstraktion Bezug auf Herstellerangaben zu den qualitativen Eigenschaften genommen.

Ein wesentlicher Kritikpunkt an der Ergebnisinterpretation ist, dass der gültige Rekonfigurationsraum aktuell kein Objekt erster Ordnung (Englisch: First Class Entity) ist. Dies gilt im vorliegenden Entwicklungsprozess bei der Beurteilung der Fehlertoleranz nur zur Entwurfszeit; zur Laufzeit sind diese Daten nur implizit verfügbar. Es gibt folglich keinen anwendungsübergreifenden Beleg, jedoch sind die erzielten Ergebnisse in Abschnitt 6.6.4.5 ein Indiz dafür, dass der Ansatz einen relevanten Ausschnitt des Rekonfigurationsraums effektiv bereitstellt.

6.7.2.6. Beurteilung der Validierungsfrage FF4.3

FF4.3 *„Treten qualitative Abstufungen zwischen adäquaten Entwurfsalternativen in einem relevanten Maße in realen Systemen auf?“*

Gegenstand der Metrik 6.6.4.6 ist die Messung der qualitativen Streuung in dem aufgespannten Rekonfigurationsraum des TET-1 ACS. Die Existenz dieser Streuungen und Berücksichtigung im realen Wartungsprozess wird im Anschluss betrachtet.

Nach vorliegender Dokumentenlage zum ACS sind die Ressourcen jeweils einzeln ansteuerbar, werden jedoch ausschließlich in unterschiedlichen Lageregelungsmodi [LHSR12] miteinander in Verbindung gebracht. Diese Einschränkung in der aktuellen Implementierung des Systembetriebs lassen keine expliziten Rekonfigurationsoptionen zur qualitativen Degradation zu. Allerdings können aus dem gleichen Dokument auch Informationen zu qualitativen Abstufungen innerhalb der Lageregelungsmodi extrahiert werden. So reduziert sich die Ausrichtungsqualität in der *groben Regelung*, vergleiche Beurteilung der Validierungsfrage FF3.2, gegenüber der *Präzisionsregelung* deutlich. Dies wird im Dokument in direkten Zusammenhang mit der Verfügbarkeit des Reaktionsradsystems gebracht. An diesem Beispiel wird deutlich, dass bereits zur Entwurfszeit des Systems eine mögliche qualitative Degradation behandelt wird.

Die Messung hat einen hohen Grad qualitativer Streuungen erhoben, sowohl graphbasiert als auch pfadbasiert. Dabei lag die Annahme vor, dass ausschließlich die Ausschlüsse auf Ressourcenebene zu Restriktionen im Variationsraumes führen. Im realen System werden diese Variationen der Ressourcenplattform und Software allerdings zusätzlich durch die Beschränkung der Lageregelung auf festgelegte Modi auf ein Minimum reduziert. Ein Vergleich bezieht sich deshalb auf dem Wegfall notwendiger Ressourcen zur Ausführung eines aktuell notwendigen Modus. In diesem Fall ist das Wartungspersonal in der Situation eine gültige Lösung über die festgelegten Modi hinaus zu identifizieren. Als anwendungsnahes Beispiel hat die Messung entlang einer exemplarischen Fehlersequenz unter anderem das ACS ohne Reaktionsradsystem untersucht. Es handelt sich um eine Variation, die aktuatorenseitig ausschließlich auf dem Magnetspulensystem basiert und dabei dennoch eine Lageregelung mit verminderter Präzision realisiert.

Im originären Entwurf des Systems sind primär keine qualitativen Abstufungen vorgesehen. Ein qualitativ gleichbleibender Betrieb wird durch einen hohen Grad an Fehlertoleranz anvisiert. Sekundär ist jedoch auch ein Notbetrieb seitens der Ressourcenplattform möglich. Diese Funktionen sind

bisher nicht im System implementiert und als Lageregelungmodus manifestiert, technisch dennoch möglich. In der Wartung kann dies ein wichtiger Ansatzpunkt für Systemerweiterungen sein. Der *DARG*-Ansatz identifiziert solche Lösungen durch die Abstraktion von Lageregelungsmodi und Fokussierung auf die tatsächlichen Ein- und Ausschlussbeziehungen innerhalb der Ressourcenplattform.

6.8. Stabilität des Konfigurationsansatzes bei variierenden Betriebsmodi

Als komplementäre Erweiterung des Ansatzes, wird in Abschnitt 5.4 die Betrachtung variierenden Betriebsmodi vorgeschlagen und konzipiert. In Anlehnung an die individuelle Validierung für drei Experimente des TET-1 wird nachfolgend die Stabilität des Ansatzes unter der Annahme variierender Betriebsmodi [MFKR18] überprüft.

Auch wenn die Relationen zwischen den *DARG*-Instanzen bereits bekannt sind, ist die Alternativenexploration aufwendig, wenn Betriebsmodi variieren und Fehler auftreten. Unter Verwendung der erzeugten Artefakte aus den werkzeuggestützen Messungen in Abschnitt 6.6, erfolgt die Validierung der Erweiterung.

6.8.1. Beurteilung optimaler Reihenfolgen von Betriebsmodi des Systems

Entsprechend Abschnitt 5.4.1 wird eine Multi-Modusanalyse auf Basis der *DARG*-Instanzen für die drei Experimente aus Abschnitt 6.6.3.1 durchgeführt.

Zunächst werden die Transitionskonfigurationen und optimalen Reihenfolgen für den fehlerfreien Fall ermittelt. Im Anschluss erfolgt eine Untersuchung unter Betrachtung einer Fehlersequenz. Für beide Varianten wird die konstante Gewichtung zur Bestimmung der Transitionswerte auf $0,33$ für w_1, w_2 und w_3 festgelegt. In der Untersuchung wird die rechenintensive

Zwischenzentralität, vergleiche Abschnitt 5.4.2, genutzt um eine umfassende Beobachtung durchzuführen.

Die Analyse berechnet individuelle Transitionswerte und untersucht deren maximalen Summen über allen möglichen Reihenfolgen von *DARG*-Instanzen hinweg. Abbildung 6.21 zeigt alle Reihenfolgen der Experimente mit den aufsummierten Transistionswerten. Jede Summe besteht aus den Transitionswerten beider Vergleiche, zum Beispiel $N1 \rightarrow N15$ + $N15 \rightarrow N7$. Die Reihenfolge $N15 \rightarrow N1 \rightarrow N7$ besitzt die höchste Summe und ist die optimale Betriebsmodussequenz im fehlerfreien Fall. Sie wird gefolgt von $N7 \rightarrow N1 \rightarrow N15$ und $N1 \rightarrow N15 \rightarrow N7$.

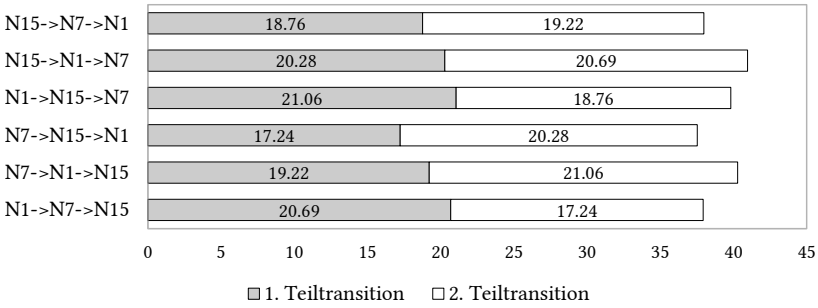


Abbildung 6.21.: Transitionswertsummen für alle Reihenfolgen

Zur Begründung des Ergebnisses wird nachfolgend die Menge der Transitionskonfigurationen zwischen einem Paar von Betriebsmodi und deren Verteilung von Transitionswerten genauer betrachtet. Eine Transition zwischen $N1$ und $N7$, vice versa, besitzt mit 86 Stück die meisten Transitionskonfigurationen. $N1 \leftrightarrow N15$ besitzt 85 und $N7 \leftrightarrow N15$ nur 77 Transitionskonfigurationen. Dadurch besitzen Reihenfolgen, die Transitionen zwischen $N1$ und $N7$ enthalten, höherer Transitionswertsummen als andere Transitionen. Die Übergänge zwischen diesen Rekonfigurationsräumen sind dabei effektiver, da sich die Räume strukturell ähnlicher sind als bei anderen Kombinationen.

Abbildung 6.22 zeigt die Verteilung der Konfigurationen für jeden einzelnen Betriebsmodusübergang. Während die Medianwerte und Wertbereiche in den Boxplots für jede Transition variieren, bleiben die Werte dennoch ähnlich

zueinander. Die Transition $N15 \rightarrow N1$ stellt beispielsweise im Durchschnitt höher bewertete Transitionskonfigurationen bereit. Diese Transition führt somit möglicherweise auch zu hoch bewerteten Konfiguration zur Umsetzung von Experiment $N1$. Die hohe Anzahl der Transitionskonfigurationen in beiden Transitionen innerhalb von $N15 \rightarrow N1 \rightarrow N7$ führt zu reduzierten Rekonfigurationskosten. Weiterhin bewirken hohe mittlere Transitionswerte adäquate Konfigurationen.

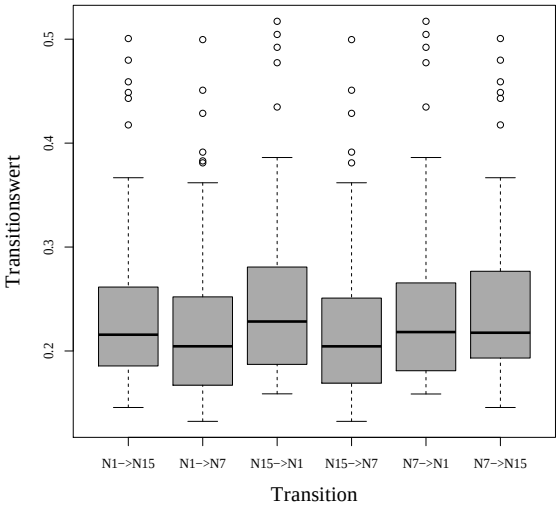


Abbildung 6.22.: Verteilung der Transitionswerte nach Transitionsrichtung ohne Annahme von Fehler

6.8.1.1. Beurteilung der Robustheit gegenüber Fehlern

Unter Einsatz der exemplarischen Fehlersequenz aus Abschnitt 6.6.3.2 wird die Robustheit der hier erzeugten Ergebnisse überprüft. In Tabelle 6.12 sind die Änderungen der Transitionswertsummen für alle Reihenfolgen und jeden Fehler aufgeführt.

Die initiale optimale Modussequenz von $N15 \rightarrow N1 \rightarrow N7$ aus der fehlerfreien Analyse bleibt für nahezu alle Fehler stabil. Mit dem Ausfall von CSS

Chipset2 Low Res wird die Transitionswertsumme jedoch durch den Wert aus $N1 \rightarrow N15 \rightarrow N7$ kurzzeitig dominiert. Ab dem Fehler von RW2 sind die Reihenfolgen indifferent zueinander. Nach der Injektion des Fehlers von MC2x sind keine weiteren Transitionskonfigurationen ermittelbar und der Rekonfigurationsprozess stoppt.

Zusammenfassend haben die Ausfälle des ONS und des CSS sowie eines Großteils der Aktuatorik des Systems die größte Auswirkung auf die Verfügbarkeit von Transitionskonfigurationen.

Tabelle 6.12.: Fehlerauswirkungen auf Transitionswertsummen

Fehler	1→7→15	7→1→15	7→15→1	1→15→7	15→1→7	15→7→1
GPS LNA1+Antenna1	36,74	39,62	37,35	38,57	**39,98**	38,04
GPS Receiver1	37,40	39,28	37,27	38,48	**39,78**	37,48
GPS LNA2+Antenna2	10,02	12,19	11,47	12,62	**12,78**	9,98
MFS Fluxgate1	10,00	12,20	11,53	12,70	**12,90**	10,05
CSS RearHead2	10,06	12,15	11,57	12,67	**12,99**	10,07
ASC DPU1 High Res	10,05	12,08	11,50	12,54	**12,94**	10,02
ASC DPU2 Low Res	10,05	12,20	11,53	12,64	**12,88**	10,02
CSS Chipset1 High Res.	10,05	12,26	11,41	12,67	**12,82**	9,96
CSS RearHead1	2,13	3,28	2,92	3,10	**3,35**	2,09
CSS FrontHead2	2,14	3,29	2,92	3,12	**3,35**	2,10
CSS Chipset2 Low Res.	2,14	3,30	2,92	3,12	**3,35**	2,10
ASC DPU1 Low Res.	1,06	2,21	2,02	**2,13**	2,22	1,06
MC1x	0,44	0,91	0,76	0,78	**0,92**	0,43
RW2	0,29	0,45	0,45	**0,46**	**0,46**	0,29
MC2y	0,29	0,45	0,45	**0,46**	**0,46**	0,29
MC2x	-	-	-	-	-	-

Zu Beginn der Fehlersequenz werden die Redundanzgruppen ausgedünnt, wobei teilweise bereits heterogene Redundanzen genutzt werden. Somit sind mit fortlaufender Fehlerinjektion die meisten alternativen Redundanzgruppen auch bereits auf eine Ressource reduziert. So fallen beispielsweise GPS Antenna1+LNA1 und CSS RearHead2 frühzeitig aus und die Fehler von GPS LNA2+Antenna2 und CSS RearHead1 können nicht über homogene Redundanzen abgedeckt werden.

Der Grund für den Wechsel der optimalen Reihenfolge nach dem Fehler von von CSS Chipset2 Low Res wird nachfolgend im Detail untersucht. Während die Menge der Transitionskonfigurationen vor dem Ausfall bereits massiv reduziert wurde, sind Transitionen aus $N1 \leftrightarrow N7$ und $N1 \leftrightarrow N15$ weiterhin zahlenmäßig (8 beziehungsweise 13) überlegen. Nach dem Fehler liegen für $N1 \leftrightarrow N15$ nur 2 und für $N7 \leftrightarrow N15$ sowie $N1 \leftrightarrow N7$ nur noch jeweils 1 Transitionskonfiguration vor. Dadurch ist der Wechsel der optimalen Reihenfolge ausschließlich auf Grundlage der Verteilung von Transitionswerten und dem höchsten Mittelwert zu erklären. Abbildung 6.23 stellt die Verteilung der Werte für beide Transitionsrichtung dar.

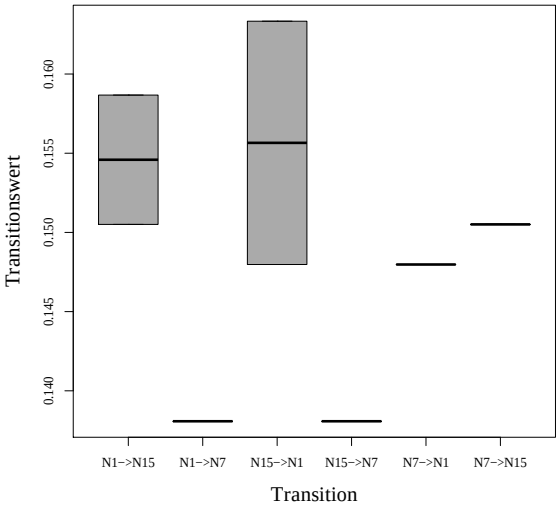


Abbildung 6.23.: Verteilung der Transitionswerte nach Transitionsrichtung mit Annahme von Fehlern

Der Boxplot zeigt, dass die Teiltransitionen von $N15 \rightarrow N1 \rightarrow N7$ und $N1 \rightarrow N15 \rightarrow N7$ gleich zueinander sind. Die ersten Teiltransitionen $N15 \rightarrow N1$ und $N1 \rightarrow N15$ besitzen einen nahezu gleichhohen Median. Für die zweiten Teiltransitionen $N15 \rightarrow N7$ und $N1 \rightarrow N7$ liegt der Median auf identischem niedrigen Level. Bezüglich des durchschnittlichen Transitionswerts sind beide Reihenfolgen somit unvergleichbar. Die größte Auswirkung auf das Endergebnis der Bevorzugung von $N1 \rightarrow N15 \rightarrow N7$ gegenüber $N15 \rightarrow N1 \rightarrow$

$N7$ hat die erste Transitionswertsumme. Diese bewirkt, dass eine optimale Gesamtsumme erreicht wird.

Zusammenfassend wird deutlich, dass $N15 \rightarrow N1 \rightarrow N7$ grundsätzlich als optimale Betriebsmodussequenz angesehen werden kann. Diese Sequenz ist bis zum 12. von 16 Fehlerfällen stabil. Zusätzlich liefert diese Sequenz den besten Trade-Off zwischen der Menge von Gemeinsamkeiten über alle Betriebsmodi hinweg. Dies hält die Rekonfigurationskosten gering. Weiterhin besitzt die Reihenfolge durchschnittliche Transitionswerte für jede Konfiguration, wodurch hohe Nutzwerte zur Erfüllung aller Experimente vorliegen.

6.9. Gesamtbeurteilung

Im Anschluss erfolgt eine Gesamtbeurteilung des vorgestellten Ansatzes entlang der Forschungsfragen dieser Arbeit. Weiterhin wird die erweiterte Anwendung der Ergebnisse diskutiert.

6.9.1. Zielerreichung der Forschungsfragen

Die Zielsetzungen gründen nach Abschnitt 1.2 auf vier zentralen Forschungsfragen. Diese wurden zu Beginn dieses Kapitels auf 17 Validierungsfragen verfeinert. Elf Fragen wurden in einer empirischen Studie untersucht, sechs Fragen durchliefen werkzeuggestützte Messungen. Die Ergebnisse der Validierungsfragen werden nachfolgend auf die Zielerreichung der jeweiligen Forschungsfrage entsprechend zurückgeführt. Dieser Abschnitt schließt den letzten Schritt des Validierungsverfahrens nach GQM-Modell, vergleiche Abschnitt 6.1.2, ab.

6.9.1.1. Beurteilung der Forschungsfrage FF1

FF1 *„Welche Faktoren limitieren die Voraussagekraft der Konfigurierbarkeit eines Systems unter Annahme kontextfreier Fehlerszenarien, festgelegter Betriebsmodi und statischer Qualitätseigenschaften?“*

Die limitierenden Faktoren der Voraussagekraft der Systemkonfigurierbarkeit unter Annahme von kontextfreien Fehlerszenarien, festgelegten Betriebsmodi und statischen Qualitätseigenschaften, werden empirisch erhoben. Nach der qualitativen Auswertung der Experteninterviews entlang der Validierungsfragen **FF1.1** bis **FF1.4** kann Folgendes geschlussfolgert werden.

Zunächst ist ein Baugruppen-übergreifender Konfigurationsbegriff festzulegen, der eine Vielzahl von Ressourcenkombinationen und Fehlerausprägungen repräsentiert. Um das Vertrauen in die autonom identifizierte Auswahl von Konfigurationsalternativen zu fördern, ist die Entscheidungsfindung umfassend zu begründen. Die architektonische Sichtweise auf das System unterstützt dies grundsätzlich, jedoch sind zusätzliche Kennzahlen zur statistischen Erläuterung notwendig.

Der hier verfolgten Ansatz abstrahiert von den technischen Details. Ressourcen sind dabei verfügbar oder persistent defekt. Entsprechend lassen sich Systemzustände in Abhängigkeit der verfügbaren Ressourcen bestimmen. Die Auswirkungen von Fehlern adressieren dabei nicht die Wechselwirkungen auf andere Ressourcen, sondern die Verringerung des Rekonfigurationsraumes. Dabei stehen vorrangig die resultierenden Folgen auf die aktuell eingesetzte Konfiguration im Vordergrund. Die Autonomie schränkt sich in der Anwendung des Ansatzes auf eine passive Beratung des Wartungspersonals ein. Dabei wird für jeden Vorschlag eines Konfigurationswechsels eine umfangreiche Begründung mit Kennzahlen und einer Visualisierung geliefert.

Bei Annahme der Kontextfreiheit von Fehlerszenarien ist sicher zu stellen, dass auch die Wechselwirkungen zwischen Ressourcenkombinationen bekannt sein müssen. Die Umwelteinflüsse können bei gegebener Flughöhe auf Konstanten abstrahiert werden, dabei existieren aber Toleranzen. Zur Steigerung der Voraussagekraft von Prognosen sind daher experimentelle Messungen erwarteter Resourcenkombination auf Teststand erforderlich, zur Reduktion von Schwellwerten in statischen Daten.

Der Ansatz liefert zur Auswahl der Ressourcen für die Testläufe eine gute Ausgangsbasis. Abweichend von der Validierung sind mit jeder Anpassung der qualitativen Werte auch Änderungen des Rekonfigurationsraum notwendig. Dies kann mit gleichbleibenden Datenmodell und Logik in einem iterativen Prozess realisiert werden.

In dem untersuchten System liegen elf Experimente als Teilmissionen der Gesamtmission zur On-Orbit-Verifikation vor. Jede Mission stellt individuell Anforderungen an die Lageregelung. Daraus wurden gemeinsame Lageregelungsmodi entwickelt.

Der Ansatz abstrahiert von dieser festen Zuteilung von Ressourcenmengen auf Lageregelungsmodi und nutzt die qualitativen Anforderungen der Missionen unmittelbar zur Auswahl adäquater Ressourcenkombinationen. Hierbei werden technische Restriktionen nicht betrachtet. Allerdings ist die Vorauswahl der möglichen Ressourcenkombinationen auch durch den Anwender veränderbar. Jede Mission wird als Benutzungsmodus der Ressourcenplattform aufgefasst und minimiert den vollständigen Rekonfigurationsraum auf eine Instanz für den Modus. Im Rahmen der Validierung des Ansatzes wurden von variierenden Betriebsmodi abstrahiert. In einer weiterführenden Studie wurde jedoch gezeigt, dass sich der Ansatz entsprechend erweitern lässt, siehe Abschnitt 6.8.

Zum Treffen von Konfigurationsentscheidungen sind redundante Datenquellen über qualitative Eigenschaften des Systems auszunutzen. So ist neben dem Aufgreifen der qualitativen Kennwerte aus Datenblättern auch die Integration der Erfahrungen von Experten und Vorgängermissionen notwendig, um eine Voraussage zu treffen. Datenblätter flugerprobter Ressourcen weisen grundsätzlich einen hohen Detailgrad auf und gelten aus zuverlässige Quelle zur Voraussage des Verhaltens. Dennoch treten teilweise lückenhafte Datenblätter auf, die mit Messwerten aus experimentellen Simulationen zu ergänzen sind. Qualitative Degradationen treten im Regelbetrieb des TET-1 nicht auf, sind jedoch technisch möglich, um einen Notbetrieb zu gewährleisten. In dem entwickelten Ansatz wird dies durch die Gewichtung der Qualitätsdimensionen und deren Mindestwerte teilweise abgedeckt. Weitere Kriterien zur Abdeckung von Geschäftsentscheidungen sind integrierbar, waren aber nicht Gegenstand der Validierung.

Die kontrollierte Abstufung von qualitativen Eigenschaften ist ein wesentlicher Beitrag des Ansatzes. So werden entsprechend der Betriebsanforderungen unterschiedliche Konfigurationsalternativen mit variierenden Nutzwerten angeboten und automatisiert, unter Berücksichtigung der Rekonfigurationskosten gegeneinander abgewogen.

6.9.1.2. Beurteilung der Forschungsfrage FF2

FF2 *„Lassen sich Fernwartungsaufwände im Betrieb durch approximative Priorisierungen von Rekonfigurationsentscheidungen zur Entwurfszeit signifikant reduzieren?“*

Die zweite Forschungsfrage stellt zur Debatte, ob sich Laufzeitaufwände in der Fernwartung durch das Approximieren und Priorisieren von Rekonfigurationsentscheidungen in der Entwurfsphase verringern lassen. Eine empirische Studie im Rahmen der Validierungsfragen **FF2.1** bis **FF2.4** richtet sich auf das Verständnis und mögliche Erweiterungen des Entwicklungs- und Wartungsprozesses. Angeschlossen daran bemessen die Metriken der Validierungsfragen **FF2.5** bis **FF2.6** die quantitativen Aufwandsänderungen unter Verwendung des Ansatzes.

Die Fernwartungsaufwände bestehen in den Bereichen Personal und der unterbrechungsfreien Kommunikation. Das Personal ist sowohl in der Diagnose als auch in der Reparatur intensiv an der Fehlerbehandlung beteiligt. Das System wechselt nach einem Fehlerfall zunächst in einen abgesicherten Betriebsmodus, der die Energieversorgung und Kommunikationsbereitschaft sichert, jedoch keine produktiven Experimente zulässt. Aufgrund einer domänenbedingten geringen Autonomieakzeptanz, werden Fehler bei der nächsten Kontaktaufnahme zum Bodensegment den Experten zur Klärung gemeldet. Durch die geringe Autonomie sind potenziell viele Diagnoseläufe notwendig, um den Fehler tiefgreifend zu analysieren. Dabei liegen die wesentlichen Beschränkungen in der Verfügbarkeit der Experten und den vordefinierten physikalischen Messpunkten, die die Fehlersymptome beobachtbar machen.

Die verbleibende Konfigurierbarkeit des Systems ist den Experten grundsätzlich bekannt. Allerdings ist dieses Wissen aktuell nur implizit aus den Housekeeping-Daten extrahierbar und bezieht sich auf momentan aktivierte Bauteilgruppen. Eine abstrakte Repräsentation als modellbasierter Konfigurationsbegriff für Analysen liegt nicht vor. Mögliche Konfigurationen werden im Vorfeld nicht explizit definiert, sondern entstehen als manuell fixierte Ressourcenkombinationen zur Erfüllung der Missionsanforderungen. Nach der manuellen Ermittlung einer Fehlerreparatur, müssen jegliche Änderungen auf Seiteneffekte mehrfach im Teststand des Systems überprüft werden, bevor eine Übertragung auf das Flugsystem stattfindet. In der Kommunikation liegt die wesentliche Herausforderung in der Kompensation weniger und

kurzer Kontaktzeiten von Satelliten und Bodensegment. Dies führt im Allgemeinen zu hohen Latenzen zwischen dem Auftreten, der Meldung und der Behebung eines Fehlers. Die erreichbare Datenrate zum Satelliten ist relativ gering, daher werden Software-Updates über mehrere Orbits verteilt.

Der thematisierte Ansatz setzt bei der Verkleinerung des sichtbaren Entscheidungsraums für Konfigurationsänderungen an. Zur Nutzung eines modellbasierten Konfigurationsbegriffs innerhalb der Analyse werden Architekturvarianten aus dem grundsätzlich architekturorientierten Quellcode abgeleitet. Somit sind Systemvariationen aus vormaligen Verzweigungen im Quellcode auch auf Architekturebene sichtbar. Anschließend wird durch eine kontinuierliche Aktualisierung der Ressourcenverfügbarkeit aus den gegebenen Housekeeping-Daten der aktuelle Systemzustand erhoben. Aufgrund des architekturbasierten Systemmodells im Bodensegment ist ein deutlich reduzierter Datenaustausch notwendig. Tritt ein Fehler auf, der eine aktuell aktivierte Ressource betrifft, liefert der Ansatz effizient eine Auswahl möglicher Lösungen. Dabei werden qualitative und strukturelle Differenzen für das Wartungspersonal aufbereitet präsentiert und eine visuelle Darstellung des Rekonfigurationsraums als kompakter Subgraph bereitgestellt. Weiterhin schlägt der Ansatz auf Basis es aktuellen Benutzungsmodus eine betriebsoptimale Konfiguration als Alternative vor. Dies ermöglicht eine schnelle Observation der Möglichkeiten und kann im Idealfall die mittlere Reparaturzeit verkürzen.

Die werkzeuggestützte Messung liefert Approximationen für die Änderungen der Personal- und Kommunikationsaufwände. Die Kommunikationskosten können durch die Abstraktion der übertragenden Diagnosedaten exemplarisch bereits für einen minimalen Fehlerfall um den Faktor 12 reduziert werden. Dies hat zwar in erster Linie nur Auswirkungen auf die genutzte Übertragungskapazität, zeigt dabei jedoch, dass das Vorhalten eines zum Satelliten synchronisierten Modells im Bodensegment, ohne weitere Kosten zu betreiben ist. Dabei ist eine sofortige Reaktion auf einen lokalisierten Fehler möglich, was wiederum den Kommunikationsaufwand in der Diagnostik verringert.

Weiterhin können personelle Kosten in Diagnose und Reparatur reduziert werden. Durch die Ermittlung möglicher hochqualitativer Konfigurationen wird dem Personal eine Vorauswahl bereitgestellt. Die zentrale Wissensbasis vereinfacht vor allem die Festlegung auf eine Konfiguration in großen

Arbeitsgruppen. In der Fallstudie konnten so über zwei Drittel der Aufwände für Abwägungen unterschiedlicher Konfigurationen eingespart werden. Weiterhin wird auch die Menge der durchgeführten Rekonfigurationen, das heißt Ressourcenänderungen bei Umschaltung, auf ein Minimum beschränkt, ohne die Rekonfigurationskosten in die Höhe zu treiben. Dadurch wird die Variantenmenge der zeit- und personalaufwendigen Testaufbauten weiterhin reduziert. Die initiale Erstellung des Modells ist zwar kostenintensiv, jedoch würden diese Kosten bei langer Systemlaufzeit kompensiert werden.

6.9.1.3. Beurteilung der Forschungsfrage FF3

FF3 *„Lässt sich die Prozesstransparenz in der Fernwartung durch den Einsatz komponentenbasierter Modelle effektiv unterstützen?“*

Neben der Senkung der Wartungsaufwände zielt der Ansatz auf eine Steigerung der Prozesstransparenz ab. In der dritten Forschungsfrage wird dabei die Effektivität des Einsatzes komponentenbasierter Modelle beurteilt. Die Metriken zu den Validierungsfragen **FF3.1** und **FF3.2** wurden simulativ bemessen. Die weiteren Fragen **FF3.3** und **FF3.4** sind Gegenstand der empirischen Studie. Die Transparenz des existierenden Wartungsprozesses ist durch Verteilung auf unterschiedliche Entwicklergruppen und Sammlungen zahlenbasierter Diagnosedaten (Housekeeping-Daten) rein technisch orientiert. Weiterhin existiert ein expliziter Konfigurationsbegriff und eine Darstellung des relevanten Rekonfigurationsraums. Neben Entwurfsdokumentationen und Fehleranalysemodellen bildet der produktive Quelltext des Flugsystems die wesentliche Basis für Fehlerdiagnosen.

Die Messung zur Anwendung des Ansatzes ergibt, dass der Rekonfigurationsraum für ein redundanzbehaftetes System modellbasiert effektiv als Graphstruktur darstellbar ist. Eine Abbildung des vollständigen Raums wäre nur in einem komplexen Graphen möglich und dabei nicht nachvollziehbar visualisierbar. Nach der Eingrenzung auf maximale Rekonfigurationskosten und der Reduktion redundanter Übergänge auf ein überschaubares Maß, liegt jedoch eine kompakte Repräsentation des Raums vor. Unter Verwendung der Laufzeitdaten zur Ressourcenverfügbarkeit erfolgt eine weitere Minimierung hin zu transparenten Subgraphen, die dem Personal zur Verfügung gestellt werden. Dabei sind Zusammenhänge zwischen Fehler und Auswirkung nachvollziehbar im Graphen darstellbar. So werden als Reaktion

auf einen relevanten Ressourcenausfall, der die aktuelle Konfiguration betrifft, konkrete Rekonfigurationspfade hervorgehoben. Insbesondere ist die Historie vergangener Rekonfigrationen neben den verbleibenen Optionen hervorgehoben. Weiterhin können zusätzlich zu einer empfohlenen Konfiguration weitere mögliche qualitativ nachgelagerte Konfigurationen optional eingeblendet werden.

Der Ansatz begründet die Vorschläge und die resultierende An- oder Abwahl diverser Ressourcen durch die strukturelle Nähe und die angestrebte qualitative Stabilität. Dabei ist die Auswahl durch den Ansatz insbesondere dann plausibel zu begründen, wenn die Rekonfigurationskosten zwischen den Alternativen hoch sind. Bei sehr ähnlichen Alternativen ist die Begründung potenziell durch die allgemeine Unschärfe der Prognosedaten unpräzise.

Empirisch wurde erhoben, dass eine Integration des Ansatzes in bestehende Wartungsprozesse möglich ist. Bezogen auf die Unterdomäne der Satellitensysteme, ist die Integration eines völlig autonomen Rekonfigurationsansatzes momentan undenkbar. Auch wenn der Ansatz eine hohe Transparenz bietet, ist die Gefahr des Systemverlustes zu hoch. Als Beratungssystem, welches Entscheidungen erhebt und nachvollziehbar begründet, jedoch nicht ausführt, wurde das Konzept grundsätzlich akzeptiert. Dabei sollte eine Interaktion mit den Experten zur Parametrisierung der Alternativensuche, zum Beispiel zur Berücksichtigung von Geschäftsentscheidungen oder „Lessons Learned“, möglich sein. In der Betriebsphase bieten prognostizierte Werte aus dem Entwurf keine hinreichende Transparenz bezüglich des Laufzeitverhaltens. Daher ist trotz der Vorauswahl durch den Ansatz, die Erprobung eines Konfigurationskandidaten auf dem Teststand notwendig.

Die architekturbasierte Dokumentation von verteilten Wartungsaktivitäten entsprechend dem Ansatz ist vielversprechend zur Aggregation von bauteilspezifischen Aktivitäten. Die graphische Notation unterstützt die Präsentation von Zusammenhängen und transparenten Erläuterungen von Rekonfigurationen und deren Folgen gegenüber den Entscheidungsträgern. Dabei müsste die Prozessintegration des Ansatzes über die Steigerung der Zuverlässigkeit argumentiert werden, da auch im Entwurf die Kostengrenzen enorm sind. Weiterhin ist sicherzustellen, dass der unpriorisierte Rekonfigurationsraum eine unüberschaubare Größe und Austauschbarkeit besitzt, die den Mehraufwand zur Analyse rechtfertigt. Entgegen dem umfassenden Einsatz innerhalb von Forschungsprojekten, ist eine Anwendung auf kommerzielle Systeme

nur in einem begrenzten Maße möglich, da gerade dort insbesondere kurze Entwurfsintervalle gefordert sind.

6.9.1.4. Beurteilung der Forschungsfrage FF4

FF4 *„Liefern meta-heuristische Suchverfahren auf Architekturbasis eine adäquate Entscheidungsbasis zur Festlegung weitreichender, nicht trivial identifizierbarer, Systemkonfigurationen mit qualitativen Abstufungen?“*

Die vierte Forschungsfrage bewertet den Einsatz von architekturbasierten Metaheuristiken zur Suche und Festlegung weitreichender Rekonfigurationen. Die Menge der Konfigurationen soll dabei nicht trivial identifizierbarer sein und qualitative Abstufungen enthalten. Zur Analyse der Validierungsfragen **FF4.1** und **FF4.3** kommt die werkzeuggestützte Messung zum Einsatz. Die empirische Studie wird zur Auswertung der Frage **FF4.2** genutzt.

Als Voraussetzung zur Anwendung qualitätsorientierter Verfahren ist zunächst erforderlich, dass die qualitativen Kennzahlen zu den Ressourcen bekannt sind. In der Domäne sind diese Daten, zumeist auf Grundlage von Flugerfahrungen, in Datenblättern manifestiert.

Weiterhin sollten qualitative Abstufungen auftreten. Die empirische Studie zeigt, dass Konfigurationen dabei grundsätzlich qualitativ und strukturell abgrenzbar sind. Im Normalbetrieb des TET-1 wird von homogenen Redundanzen ausgegangen, die strukturell oder qualitativ nicht fassbar sind. Werden die Missionsziele jedoch weniger Ressourcen-restriktiv aufgefasst, sind auch Lösungen mit Variationen von Ressourcenmengen und schwankenden Qualitäten möglich. Technisch ist dies umsetzbar, jedoch nur für den Notbetrieb des Systems implementiert. In der erweiterten Umsetzung der Fallstudie werden diese Restriktionen verringert und Ressourcenkombinationen mit unterschiedlichen Nutzwerten erlaubt. Somit ist eine qualitative Degradation im Lösungsraum existent.

Die Bewertung der Adäquatheit des metaheuristischen Ansatzes erfolgt auf Basis eines theoretischen Vergleichs, zu der erschöpfenden Suche im synthetisch generierten vollständigen Rekonfigurationsraums des TET-1. Ein direkter Vergleich zur Praxis ist nicht möglich, da dort aktuell kein Konfigurationsbegriff vorliegt und somit der Raum nicht explizit verfügbar ist.

Die Messungen zeigen, dass das metaheuristische Verfahren einen relevanten Ausschnitt des Entwurfraums liefert, da die kostenorientierten Minimierungen den Raum zwar verkleinern, jedoch die qualitative Priorisierung zugleich eine sinnvolle Auswahl von Konfigurationen gewährleistet. Im vollständigen Graphen sind die Rekonfigurationskosten sogar höher, da zunächst gierig[34] hoch-qualitative Konfigurationen gewählt werden. Anschließend ist jeodoch kostenintensive Wechsel in minder-qualitative Varianten notwendig, um weitere Fehler zu behandeln. Aufsummiert konnten nach beiden Verfahren identische Gesamtqualitäten für die Fallstudie ermittelt werden. Bei der Auswahl von alternativen Konfiguration liegen zudem Varianten mit unterschiedlichem Ressourceneinsatz und Nutzwerten vor, die jedoch die qualitativen Anforderungen des jeweiligen Benutzungsmodus einhalten. Dies zeigt, dass auch nach der Minimierung noch eine hohe qualitative Streuung in den Nutzwerten der verbleibenden Konfigurationen existiert.

6.9.2. Diskussion

Aus der Beurteilung der Forschungsfragen gehen mögliche Beschränkungen und Anschlusspunkte für Folgearbeiten hervor. Limitierungen treten vorrangig bei Investitionen im Entwicklungsprozess und der Anwendbarkeit architekturbasierter Analysen auf. In Bezug auf die Wiederverwendung von Entwurfsmodellen und die Steigerung von Interaktionen mit Experten sind Erweiterungen absehbar.

6.9.2.1. Limitierung: Steigende Investitionen im Systementwurf

Die Entwicklung eines Raumflugkörpers ist kostenintensiv, standardisiert und durchläuft eine Vielzahl von Iterationen aus Entwurf und Qualitätssicherung. Bereits für Kurzzeitmissionen, wie dem TET-1, ist diese Phase sehr komplex und langwierig, um einen zuverlässigen Betrieb über die anvisierte Laufzeit und darüber hinaus zu gewährleisten. In kommerziellen Missionen wird der Entwurf stark gestraft, um möglichst schnell in einen rentablen Betrieb des Systems überzugehen, vergleiche Tabelle D.43. Bei Forschungsmissionen sind zwar grundsätzlich mehr Freiheiten für innovative Lösungen

[34] Im Sinne eines Greedy-Algorithmus

vorgesehen, dennoch liegt das hauptsächliche Augenmerk weiterhin auf der Wiederverwendung bekannter Lösungen. Zur Anwendung des Ansatzes muss daher ein deutlicher Zugewinn in der Zuverlässigkeit in späteren Betriebsphasen ersichtlich sein. Sofern zusätzlich das Budget verfügbar ist und das Projektmanagement zustimmt, siehe Tabelle D.47, können die Entwicklungsaufwände gesteigert werden. Ein wesentliches Argument ist, dass der Ansatz eine umfangreiche Analyse der Beziehungen im unsortierten Entwurfsraum vornimmt und dabei vormals unberücksichtigte Konfigurationen aufgedeckt. Dies ermöglicht es Engpässe in der Redundanzwahl zu identifizieren und somit Ausfallrisiken zu verringern. Auf dieser Grundlage können die Ressourcenauswahl und zugehörige Implementierung angepasst und die Zuverlässigkeit des Systems gewährleistet werden.

6.9.2.2. Limitierung: Unschärfe frühzeitiger qualitativer Analysen des erzeugten Rekonfigurationsraums

Die Vorberechnungen zur approximativen Abwägung von Konfigurationsalternativen unterliegt einer natürlichen Unschärfe. Die Domäne lässt zwar die Annahme von statischen Werten für Ressourceneigenschaften und Umweltbedingungen zu, jedoch nur bis zu einem gewissen Maße der Produktreife. Die Beurteilung ohne Berücksichtigung von dem Ausführungskontext, ist vor allem durch die Abstraktion der zeitlichen Dimension unscharf. So wird beispielsweise die Sonnenkamera des TET-1 bei Messungen im Erdschatten keine plausiblen Daten liefern. Eine grobe Vorauswahl ist bei großen Streuungen von theoretisch erreichbaren Nutzwerten vorausberechenbar und kann dabei durch Obergrenzen von Rekonfigurationskosten weiter reduziert werden. Die konkrete Auswahl einer einzelnen Konfiguration ist durch den Ansatz zwar möglich, wird in der Praxis aber durch Toleranzen in den Prognosewerten gestört.

6.9.2.3. Limitierung: Spezifische Ressourcenanforderungen

Weiterhin ist eine rein qualitätsbezogene Auswahl von Ressourcenkombinationen zu unspezifisch für existierende Anforderungsprofile. Für hochspezialisierte Betriebsmodi sind auch die Spezifikationen an die Ressourcen sehr eng gefasst. Dies ist zum Beispiel im TET-1 durch feste Zuordnung

von Ressourcen, Lageregelungsmodi und Missionen (Benutzungsmodi) strikt vorgegeben. Auch wenn eine theoretische Austauschbarkeit zwischen heterogenen Ressourcen vorliegt, erfüllt diese aktuell nicht die Missionsanforderungen. Dennoch werden diese strengen Anforderungen bereits zur Umsetzung eines Notfallbetriebs teilweise gelockert. Dementsprechend sind weiterführende Ansätze vielversprechend. Eine stärkere Orientierung auf qualitative Eigenschaften zu der Ressourcenauswahl zur Laufzeit bedarf der Integration universeller Sensorik und Aktuatorik, die in heterogenen Redundanzgruppen trotz unterschiedlicher Umweltbedingungen ähnliche Ergebnisse liefern. Dies widerspricht zwar noch der Spezialisierung von heutiger Hardware für Raumflugkörper, kann aber die Verfügbarkeit und Lebensdauer eines Systems signifikant steigern. Dabei sind Missionsanforderungen um qualitative Degradierungen zu erweitern, die bereits ab der Entwurfsphase eine hohe Rekonfigurierbarkeit auf Basis heterogener Redundanzen fördern.

6.9.2.4. Erweiterung: Entkopplung von Entwurfsmodellen und Artefakten in der Betriebsphase

Der existierende Wartungsprozess verweist nur zur Dokumentation auf Entwurfs- und Analysemodelle aus der Entwurfsphase. Eine feste Einbindung und kontinuierliche Anpassung der Modelle erfolgt nicht. Eine Kopplung von produktivem Quellcode und abstraktem Architekturmodell erfordert eine stärkere Ausrichtung zum modellbasierten Software Engineering. Wird eine architekturzentrierte Entwicklung verfolgt, können Erkenntnisse aus der Alternativenabwägung bereits frühzeitig in eine zentrale Datenbasis überführt werden. Dieses Wissen aus der Entwurfszeit wird im Sinne einer *Knowledge Carrying Software* [GRG+14] für die Betriebsphase verfügbar gemacht. Die produktive Implementierung des Systems wird im Betrieb dann um ein kontinuierlich synchronisiertes Modell ergänzt. Dies steigert die Sichtbarkeit von Historie und Folgen von Rekonfigurationen für das Personal in Entwicklung und Wartung.

6.9.2.5. Erweiterung: Interaktionen mit Experten

In späteren Entwurfsphasen ist zusätzliches Wissen in die Datenbasis zu integrieren, um die Entwurfs- und Geschäftsentscheidungen der Entwickler

in späteren Rekonfigurationen zu berücksichtigen. Letztlich erstreckt sich dieser Bedarf an Interaktion auch bis in die Betriebsphase. Eine feingranulare Abwägung von einer Untermenge von Konfigurationen ist nur unter Berücksichtigung der Laufzeitdaten möglich. Die Integration von zusätzlichen Daten zur Laufzeit erfordern eine Erweiterung des Ansatzes. Zur effizienten Umsetzung iterativer Analysen des Rekonfigurationsraums ist es notwendig, dass Analyseergebnisse vorheriger Iterationen, insbesondere aus der Entwurfsphase, berücksichtigt werden. Dabei kann die zunehmende Reduktion des Raums an Konfigurationsalternativen und der aktuell selektierte Subgraph ausgenutzt werden. Weiterhin können Iterationen auf ausschließlich auf die geänderten qualitativen Dimensionen beschränkt werden. Somit ist zwar die Aggregation von Nutzwerten und die nachgelagerte Kantenpriorisierung im Subgraphen zu wiederholen, die Analyse der Rekonfigurationskosten und die damit verbundene Kantenreduktion bleibt jedoch erhalten.

7. Verwandte Arbeiten

Der Ansatz dieser Arbeit ist im Bereich der Analyse von variantenreichen Systemen einzuordnen und bietet ein meta-heuristisches Verfahren zur Exploration innerhalb eines dynamischen Rekonfigurationsraums. Dabei werden Distanzen zwischen möglichen Alternativkonfigurationen untersucht und eine Entscheidungsunterstützung in deren Auswahl gegeben. Der Ansatz identifiziert Wartungsaktivitäten und beschreibt Rationale für den Konfigurationswechsel. In der Literatur liegen Anknüpfungspunkte in dem Stand der Technik und ähnlichen Arbeiten vor.

7.1. Software Engineering in der Raumfahrt

Zur thematischen Einordnung wird der Stand der hier relevanten Techniken in der Raumfahrtdomäne untersucht. Neben der Rolle des modernen Software Engineerings wird der Wartungsprozess in dem Bereich betrachtet und mit dem entwickelten Ansatz abgeglichen.

7.1.1. Modellbasierte Ansätze in Entwurf und Betrieb

Neben dem langjährigen Einsatz modellbasierter Entwicklungs- und Analyseprozess im Entwurf von Raumflugkörpern, findet die Anwendung solcher Techniken zuletzt auch verstärkt Einzug in die Softwaresysteme für Bodensegmente [WPCE17]. Neue Infrastrukturen für moderne *Monitoring & Control*-Anwendungen werden geschaffen, um aktuelle Systeme, vergleiche SCOS-2000[1], zu ersetzen. Um der fortschreitenden Softwarealterung entgegenzuwirken hat die ESA (Englisch: European Space Agency) die Initiative EGS-CC (Englisch: European Ground Systems - Common Core) gegründet.

[1] `https://www.esa.int/Our_Activities/Operations/gse/SCOS-2000`

Eine zukünftige Generation der Bodensysteme deckt damit die gesamte Betriebsphase eines Raumflugkörpers ab. Die Initiative greift dabei auch in den Entwurf ein. Die wesentlichen Ziele liegen in dem nahtlosen Übergang von Montage, Integration und Testen (AIT, Englisch: Assembly, Integration and Testing) hin zum Betrieb [PC14]. Neben der Modernisierung der Softwaresysteme soll auch der Austausch zwischen Prozessbeteiligten vereinfacht werden.

Der in dieser Arbeit entwickelte Ansatz kann im Zuge der Modernisierung der Bodensysteme in den Wartungsprozess integriert werden. So dient der Ansatz als synchronisiertes Modell des Flugsystems im Bodensegment. Dabei repräsentiert das Modell den momentanen Zustand des Systems, beschreibt dessen Änderungshistorie und dient der Unterstützung von Wartungsaufgaben.

Weiterhin definiert die ESA in dem Dokument *ECSS-E-TM-10-23A*[2] Empfehlungen für einen modellbasierten Datenaustausch zur Unterstützung des Entwicklungsprozesses. Dies beinhaltet die Erfassung der Daten in gemeinsame Datenbanken und deren Bereitstellung im gesamten Lebenszyklus des Systems.

Die Erfassung und Persistierung aller Entwicklungs- und Wartungsdaten ist auch ein Ziel des Ansatzes dieser Arbeit. Der deterministische Architekturrelationsgraph bietet dabei ein Modell, um den Zustand des Rekonfigurationsraums präzise zu beschreiben. Aus diesem Modell lassen sich in der Wartung weitere Kennzahlen ableiten.

Neben der Telekommandierung (Steuerung) der Flugsysteme, steigen auch die Aufwände in der Auswertung der empfangenden Telemetriedaten über den Zustand des Systems. Zur Bewältigung der zugleich ansteigenden Datenmenge in langjährigen Missionen und gleichzeitigem Abwandern der Experten zu Folgeprojekten, werden intelligente Techniken des Software Engineerings eingesetzt [EMKS+17]. Statische Annahmen über Messtoleranzen sind für langjährige Missionen nicht möglich, da sich die Betriebsbedingungen aus einer Vielzahl veränderlichen Parameter ergeben. Mittels Data Mining werden einerseits Muster in den Daten erhoben und zugleich Wer-

[2] ECSS-E-TM-10-23A: Space system data repository, ergänzend zu ECSS-E-ST-10C Revision 1, `http://ecss.nl/hbstms/ecss-e-tm-10-23a-space-system-data-repository`

tetoleranzen evolutionär angepasst [EMKS+17]. Dies vermeidet eine hohe Rate an falsch-positiv gemeldeten Intervallverletzungen.

Der verfolgte Ansatz reduziert den notwendigen Datenaufwand in der Telemetrie und Telekommandierung. Ziel ist es die Menge der übertragenden Daten auf ein Minimum zu beschränken, um die Laufzeitinstanz des deterministischen Architekturrelationsgraphens synchron zum Flugsystem zu halten.

Weiterhin schreitet die frühzeitige Qualitätsorientierung der Softwareentwicklung für die Flugsysteme voran. Dabei wird als ganzheitliches Qualitätsattribut die Softwareresilienz in Aussicht gestellt. Softwareresilienz ist die Fähigkeit eines Systems einen technischen Defekt selbstständig zu kompensieren, um einen Totalausfall zu vermeiden. Eine Literaturstudie im Bereich des Software Engineerings über mehr als 50 Satellitensysteme aus sechs Raumflugprogrammen [PMS18] ergab, dass Softwareresilienz in der Domäne noch nicht als festes Qualitätsattribut etabliert ist. Die Intensität der Software nimmt zwar in diesen Systemen zu, jedoch beziehen sich die wesentlichen Methoden der Resilienz auf Hardwareressourcen. Die Autoren stellen jedoch heraus, dass eine Entwicklung in der Domäne absehbar ist, unter anderem durch die systemweite Inspektion flugerprobter Komponenten.

Der vorgestellte Ansatz adressiert vor allem die Ausnutzung heterogener Redundanzen im System. Dadurch lässt sich die Zuverlässigkeit des Systems, innerhalb der verfügbaren Konfigurationen, optimieren. Der Ansatz ermittelt Rekonfigurationspfade und versucht für jeglichen Zustand der Ressourcenplattform eine betriebsoptimale Konfigurationsalternative zu ermitteln.

In der Raumfahrtdomäne wird größter Wert auf die Flugerprobung von Hardware- und Softwareelementen gelegt [BBN10] . Generell gilt, dass eine flugerprobte Komponente beziehungsweise Ressource als zuverlässig gilt, sofern sie sich in Klassifizierungsverfahren bewährt hat. Die Klassifizierungen gleichen sich in allen prominenten Raumfahrtagenturen. Für den europäischen Raum greift die ESA auf den Standard ECSS-E-ST-10-02C[3] mit vier Kategorien zur Beurteilung eines erprobten Systems zurück. Der hohe Grad der Wiederverwendung zeigt deutliche Vorteile in der Verifikation und Zuverlässigkeit. Die Geschichte zeigt jedoch, dass die Kompatibilität wiederverwendeter Bauteiltypen auch zu katastrophalen Folgen führen kann.

[3] Fassung vom 1. Juni 2012, `http://ecss.nl/standard/ecss-e-st-10-03c-testing/`

So waren beim fehlgeschlagenen Jungfernflug der Trägerrakete *Ariane 5* inkompatible Berechnungen eines Bauteiltyps der Vorgängerrakete durchführt worden. Die Sonde *Mars Observer* nutze in einem Bauteil Berechnungen, die sich auf einen niedrigen Erdorbit bezogen [Hei14]. Ein Totalverlust des Systems war jeweils die Folge. Eine Studie zeigt, dass ein möglicher Grund in der vorrangigen Einzelbeurteilung der Bestandteile besteht [Hei14]. Eine umfassende kompositionelle Betrachtung in Abhängigkeit zu dem verbauten System wird anvisiert.

In dieser Arbeit werden die funktionalen Aspekte der Konfigurationen entsprechend der Annahmen in Abschnitt 1.3.2 ausgeklammert. Sämtliche Konfigurationen im deterministischen Architekturrelationsgraph gelten als funktional korrekt spezifiziert. Die Validität einer Konfiguration wird ausschließlich von der Verfügbarkeit innerhalb der Ressourcenplattform beeinflusst.

In der Kompensation von Fehlern treten in heutigen Raumflugkörpern qualitative Degradierungen auf, die Missionsanforderungen sind jedoch an feste qualitative Anforderungen geknüpft. Im normalen Betrieb wird so anstelle von analytischer Redundanz, vergleiche Abschnitt 2.1.8, auf Baugruppen von identischem Typ und technischen Spezifikationen zurückgegriffen. Zur Verlängerung der Laufzeit prominenter Raumflugkörper kam es jedoch zunehmend zu Abweichungen dieser strikten Vorgabe. Neben der Reduktion der Missionsziele des *Kepler Teleskops*, vergleiche Abschnitt 1.1, steht auch eine nachträgliche Anpassung der Degradierungensstufen des populären Röntgenteleskops *XMM-Newton* bevor. Im originären Systementwurf war der Betrieb mit drei Reaktionsrädern unumstritten, ein viertes Rad wurde ausschließlich redundant verbaut. Nach zwölf Jahren Missionszeit wird inzwischen eine Inbetriebnahme des Rads untersucht [Pan12]. Aktuelle Berechnungen zeigen, dass durch die langsamere Rotation aller Räder langfristig bis zu 50% Treibstoff für die Entsättigung des Drehmoments gespart werden kann.

Der entwickelte Ansatz setzt die Existenz von Degradierungsstufen voraus, um heterogene Redundanzbeziehungen trotz qualitativer Schwankungen zuzulassen. Die Validierung in Abschnitt 6.6 zeigt, dass sich in einem praxisnahen Szenario der Rekonfigurationsraum somit deutlich steigern lässt.

7.1.2. Wartung fehlertoleranter Raumfahrtsysteme

Der Wartungsprozess von heutigen Raumfahrtsystemen wird durch etablierte Modelle, domänenspezifische Sprachen und Werkzeuge zur Fehlererkennung und -behebung unterstützt, vergleiche Abschnitt 2.1.8.

Die NASA (Englisch: National Aeronautics and Space Administration) nutzt Techniken des Software Engineerings zur Qualitätssicherung in Entwurf und Betrieb fehlertoleranter Raumfahrtsysteme [SFFW15]. Eine besondere Rolle kommt dabei dem Qualitätskriterium der Wartbarkeit (Englisch: Maintainability) zu [CDL15]. Diese spaltet sich in die Unterkategorien Zuverlässigkeit, Verfügbarkeit und Instandhaltbarkeit (RAM, Englisch: Reliability, Availability, Maintainability) auf. Hierbei werden vor allem technische Schnittstellen so entworfen, dass in der Wartung die notwendigen Daten zur Fehleranalyse erhoben werden können.

Innerhalb dieser Arbeit wird vorrangig der Wartungsprozess unterstützt, wobei eine inhärente Wartbarkeit des Systems angenommen wird. Erst wenn ein großer Raum von möglichen Konfigurationen besteht, kann der Ansatz diese Auswahl nutzen, um betriebsoptimale Konfigurationen zu identifizieren.

In der Implementierung der Betriebssoftware von Raumflugkörpern finden N- beziehungsweise Multi-Versionstechniken (NVP, Englisch: n-version programming) [Avi67, Lyu95] zur Fehlerbehandlung Anwendung [CJCG08, DL94]. Mehrfache Implementierungen mit leichten Abweichungen in der Funktionalität, jedoch Ähnlichkeiten im funktionalen Kern, werden für die Entscheidungsfindung bereitgestellt. Die Erprobung jeglicher Konfigurationsalternativen verbleibt aktuell bei dem Entwicklungspersonal, um einen hohen Grad an Qualitätssicherung in der Domäne der Weltraumsysteme zu sichern. Weiterhin steht NVP in der Kritik, da die Multiplikation homogener Redundanzen im System oftmals auch Spezifikationsfehler multipliziert, vergleiche empirische Studie von Knight und Leveson [KL86].

Die Exploration möglicher Konfigurationsalternativen und die explizite Darstellung der Beziehungen dieser, liegt im Fokus des vorgestellten Ansatzes. Analog zu den eingesetzten Versionstechniken, ist das Ziel, eine möglichst umfangreiche Auswahl an Konfigurationen zur Umsetzung eines einzelnen Betriebsmodus vorzuhalten. Der Ansatz nutzt dabei jedoch ein variantenreiches gemeinsames Ausgangsmodell und vermeidet dadurch eine

Vervielfachung des Entwicklungsaufwands in der Konfigurationserzeugung und damit verbundenen Qualitätsschwankungen in Entwicklungsartefakten. Dabei werden insbesondere heterogene Redundanzen ausgenutzt, um eine große Vielfalt innerhalb der Konfigurationsalternativen sowie einen hohen Grad an Rekonfigurierbarkeit zu bewirken.

Moderne Ansätze unterstützen den Prozess dabei durch ein kooperatives Engineering von Systemimplementierung, Entwurfsmodellen und reichhaltigen Analysemodellen. Der Ansatz *COMPASS* [EKN+12] ermöglicht durch einen konsistent modellbasierten Entwicklungsprozess, die formale Verifikation der Betriebssoftware von Flugsystemen. Neben einer domänenspezifischen Modellierungssprache auf Basis der AADL [FGH06], die auf die technischen Anforderungen zugeschnitten ist, wird eine Werkzeugumgebung zur semiautomatisierten Verifikation angeboten, siehe Abbildung 7.1.

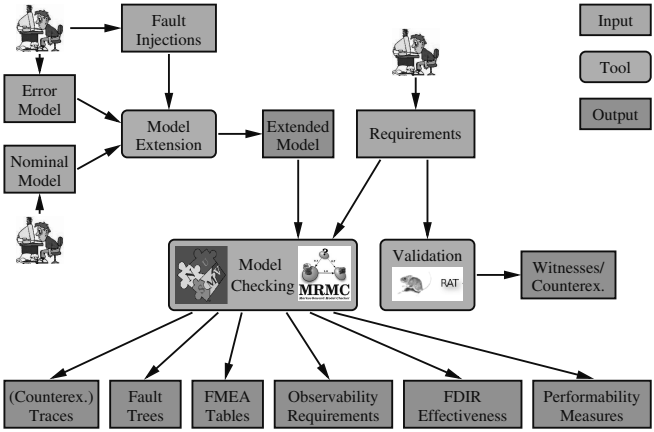


Abbildung 7.1.: Überblick COMPASS-Werkzeugkette (Quelle: [Nol15])

Durch den Einsatz etablierter Ansätze [CCG+02] aus dem Bereich der Modellüberprüfung (Englisch: Model Checking) werden zur Entwurfszeit Fehleranalysemodelle erzeugt und eine funktionale Korrektheit der Software belegt. Noll [Nol15] fasst den Ansatz zur Evaluierung systemweiter Korrektheit, Betriebssicherheit und Leistungsfähigkeit zusammen. Dabei stehen die Beurteilung einzelner Variationen der Software im Vordergrund.

Analog zur kompositionellen Beurteilung von flugerprobten Systeme, liegt auch die Verifikation neuartiger Implementierungen nicht im Fokus dieser Arbeit. Die funktionale Korrektheit der Konfigurationen wird vorausgesetzt. Weiterhin ist der Konfigurationsbegriff der Betriebssoftware auf einer höheren Abstraktionsebene angelegt.

7.2. Explizite Modellierung und Analyse von Variabilität in Softwaresystemen

Ein variantenreicher Entwurfsraum bildet in dem vorgestellten Ansatz die Basis für die Relationsbildung zwischen möglichen Konfigurationen. Nach dem Prinzip einer Feature-orientierten Domänenanalyse (FODA) [KCH+90] werden zunächst die Variabilität in dem betrachteten Software-/Hardwaresystem als Produktlinie untersucht. Dabei werden Informationen aus der Analyse der Eintrittswahrscheinlichkeiten und Auswirkungen von Ressourcenfehlern genutzt, um ein Feature-Modell aufzubauen. Auf Basis dieser Analyse werden Entwurfsmodelle um Variationspunkte erweitert. Dies definiert den Problemraum für den vorgestellten Ansatz und wird dem Domain Engineering [CE00] zugeordnet. Auf Grundlage der expliziten Variabilität in den Entwurfsmodellen wird ein Teil des möglichen Entwurfsraums erzeugt und qualitativ vorselektiert. Abschließend wird im Application Engineering der priorisierte Rekonfigurationsraum als Lösungsraum mit einer Menge von qualitativ bewerteten Produktvarianten erzeugt, vergleiche [PBDL05].

FODA bietet Möglichkeiten zur Analyse expliziter Variabilität im Problemraum. Der präsentierte Ansatz nutzt diese Methoden innerhalb der Vorausberechnung des Rekonfigurationsraums, der für eine Teilmenge der insgesamt möglichen Varianten des Systems zusätzliche Informationen zu den Beziehungen zwischen den Varianten bietet. Anders als in einem Feature-Modell werden diese Beziehungen durch einen Relationsbegriff beschrieben. Relationen steuern dabei die Auswahl zwischen Varianten anhand deren strukturellen und qualitativen Ähnlichkeiten. Sonstige Analysen der expliziten Variabilität im Problemraum sind nicht Gegenstand dieser Arbeit.

Der Ansatz dieser Arbeit hat starke Bezüge zu delta-orientierten Transformationen innerhalb von Entwicklungsartfakten von Schaefer et al. [SBB+10].

Mit Bezug auf das komponentenbasierte Software Engineering definieren Lochau et al. [LLL+14] eine Architekturbeschreibungssprache zur Modellierung delta-orientierter Architekturen. Auf Grundlage einer vorgegebenen Kernarchitektur definieren Delta-Operationen das Hinzufügen, Entfernen oder Ändern von Komponenten und Konnektoren. In diesem Zusammenhang nutzten Lachmann et al. [LLL+15] Delta-Graphen zur Beschreibung und Identifikation struktureller Änderungen zwischen Produktvarianten einer Produktlinie. Abbildung 7.2 zeigt einen Delta-Graphen für drei Architekturvarianten.

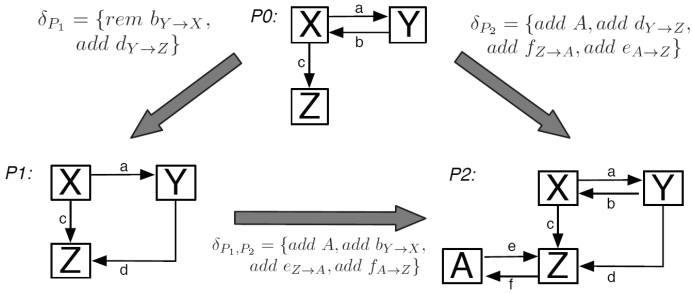


Abbildung 7.2.: Delta-Graph für drei Architekturvarianten (Quelle: [LLL+14])

Neben der Beschreibung der Änderungen, bietet der verwandte Ansatz die Möglichkeit, Komponenten auf Basis der Änderungen ihrer Ein- und Ausgangskanten zueinander zu gewichten. Lachmann et al. nutzen diese Eigenschaft zur Priorisierung von Testfällen und daher auch indirekt zur An- oder Abwahl von Architekturausprägungen.

Im Vergleich zu dem Ansatz dieser Arbeit werden strukturelle Änderungen innerhalb der Softwarearchitektur betrachtet. Jedoch adressieren Delta-Graphen rein strukturelle Unterschiede in den Schnittstellen der Komponenten. Eine Beurteilung der qualitativen Aspekte findet nicht statt. Weiterhin werden weitreichende Rekonfigurationen über mehrere Architekturvarianten hinweg nicht betrachtet. Der in dieser Arbeit verfolgte Ansatz deckt beide Aspekte. Delta-Graphen ergänzen den Ansatz in der möglichen Betrachtung von Änderungen auf Ebene der Betriebssoftware. Entgegen der ausschließlichen Ausrichtung auf Ressourcenänderungen innerhalb einer

Ressourcenplattform wären somit auch Aufwände für Softwareanpassungen in die Entscheidungsfindung für Rekonfigurationen einbeziehbar.

Neben der Betrachtung der Variabilität zur Entwurfszeit, erlauben dynamische Produktlinien [HHPS08] die Beschreibung evolvierender Modellen zur Laufzeit. Dabei findet eine Ko-Evolution der Variabilitätsmodells (Problemraum) und des Systemmodells (Lösungsraum) statt. Insbesondere in der Wartung können so nachträgliche Änderungen am System in das Variabilitätsmodell eingepflegt werden. Dies bewahrt die Konsistenz beider Artefakte und ermöglichst weitere Analysen. Dynamische Produktlinien sichern somit die Variabilität und deren Analysierbarkeit, nachdem Änderungen im Lösungsraum vorgenommen wurden [QRV+15].

Der in dieser Arbeit vorgestellte Ansatz beschreibt einen dynamischen Ausschnitt der expliziten Variabilität als erreichbarer Rekonfigurationsraum des Systems. Durch eine variable Ressourcenplattform nimmt die Variabilität in einem deterministischen Architekturrelationsgraphen zunehmend ab. Der Ansatz beschreibt dies mit einem fallenden Grad an Rekonfigurierbarkeit. Ergänzungen neuer Ressourcen sind aktuell nicht vorgesehen, entsprechend wächst der Raum während der Laufzeit nicht an. Eine Anpassung des Problemraums ist durch eine Rückführung der zur Laufzeit noch verfügbaren Rekonfigurationsoptionen möglich. So können in dem Variabilitätsmodell die Optionen zur Komponentenauswahl angepasst werden. Eine dynamische Produktlinie bietet grundsätzlich die gleichen Mechanismen, liefert aber keine Priorisierung der Produktvarianten untereinander. Eine explizite Relationsbildung zwischen Varianten liegt somit nicht vor. Weiterhin erfasst der Ansatz dieser Arbeit den zeitlichen Verlauf der Variabilität und bietet somit eine Historie über erfolgte Rekonfigurationen.

Garcia et al. [GPEM09] beschreiben „Bad Smells“ auf Architekturebene für Systeme ohne Variabilität. Bad Smells sind generelle Indikatoren in Entwicklungsartefakten für schlechte Entwurfs- oder Implementierungsentscheidungen [FBB+99]. So lassen sich unter anderem irrelevante beziehungsweise redundante Konnektoren oder mehrdeutige Schnittstellen in Architekturmodellen identifizieren. Auch in der statischen und der dynamischen Produktlinienforschung treten nach de Andrade et al. [AAC14] Bad Smells innerhalb der Variabilität auf. Durch solche Fehlentscheidungen in der Variabilitätsmodellierung können inadäquate Produktvarianten aus einer Kernvariante

erzeugt werden. Funktional ist jede Variante zwar valide, weist jedoch Schwächen in den Bereichen Verständlichkeit, Testbarkeit, Erweiterbarkeit oder Wiederverwendbarkeit auf. Insgesamt schaden alle Punkte letztlich der Wartbarkeit einer Variante. Neben den architektonische Bad Smells, führen die Autoren insbesondere einen Smell ein, der auf eine hohe Konzentration von Features in Architekturkomponenten hinweist.

Die verwandten Arbeiten untersuchen strukturelle Qualitätsmerkmale der Wartbarkeit innerhalb (variantenreicher) Architekturen. Das Wissen über Bad Smells wird im Entwurf erhoben, um spätere Wartungsaufwände zu reduzieren. Dabei setzen die Ansätze im Bereich der Refaktorisierung von Architekturentwürfen an. Die Arbeit unterstützt die Wartbarkeit der erzeugten Architekturvarianten, bietet jedoch keine Unterstützung für den Wartungsprozess. Hier setzt der Ansatz in dieser Arbeit ein, um qualitative Bewertungen von Architekturen dafür zu nutzen, unterschiedliche Architekturvarianten voneinander zu unterscheiden. Beide Arbeiten können jedoch die statische qualitative Architekturbewertung in dem hier verfolgten Ansatz ergänzen. So wäre neben der ressourcenbasierten Erhebung von Nutzwerten, auch eine frühzeitige Berücksichtigung der Wartbarkeit möglich. Hierdurch ließen sich einerseits schlecht wartbare Konfigurationen im Vorhinein ausschließen, sowie andererseits eine Priorisierung von Alternativkonfigurationen entlang deren Wartbarkeit durchführen.

7.3. Meta-heuristische Erzeugung und Exploration des Problemraums

Neben der expliziten Beschreibung der Variabilität, existieren diverse suchbasierte Ansätze zur Validierung des erzeugten Variantenraums, vergleiche [LHLE15]. Dabei werden zumeist genetische Algorithmen eingesetzt. Die Generierung von Produkten aus Feature-Modellen fokussiert sich in der Literatur stark auf den Bereich des Softwaretestens, zum Beispiel [JSE96, PHP99, SK09, HPP+13, BLLS14]. Einige Ansätze sind mit dem hier genutzten Ansatz verwandt.

Garvin et al. [GCD10] kombinieren eine evolutionäre Suche mit der Feature-Modellierung. Unter der Verwendung von „Simulated Annealing“ erweitern

die Autoren einen Algorithmus zur Testfallgenerierung zur Bestimmung von gültigen Feature-Konfigurationen unter Vermeidung eines kombinatorischen Aufwands. Auf Basis einer tabellenartigen Repräsentation (Bit-Vektor) des Feature-Modells, selektiert der Algorithmus paarweise Features. Nach jeder Änderung der Selektion wird die Erfüllbarkeit der erzeugten Variante erprobt. Eine Fitness-Funktion maximiert dabei die Abdeckung von Feature-Paaren. Im Anschluss wird die Schnittmengen über alle maximierten Paare gesucht, um die Auswahl der Testfälle abzuschließen. Ähnlich zu diesem Ansatz wenden Ensan et al. [EBG12] einen Genetischen Algorithmus zur Erzeugung von Produktvarianten aus einem Feature-Modell an. In beiden Ansätzen repräsentiert jedes Gen in einem Chromosom ein Feature. Die Fitness einer Produktvariante richtet sich nach einer Evaluation der Feature-Auswahl und deren Nebenbedingungen innerhalb des Feature-Modells. Wenn ein Produkt invalide ist, wird jeweils eine Mutation des Chromosoms durchgeführt.

In dem in dieser Arbeit vorgestellten Ansatz beschreibt ein Feature-Modell die Variationspunkte im Systemmodell und beschränkt die Ausprägungen von Konfigurationen durch ressourcenbedingten Ein- oder Ausschlussbeziehungen. Der Ansatz betrachtet dabei keine Abdeckungskriterien, sondern sucht Konfigurationen mit minimalen strukturelle Distanzen zueinander. Weiterhin wird angenommen, dass die Anzahl der Variationspunkte durch Ressourcenfehler reduziert wird und betroffene Konfigurationen über die Zeit invalide werden können.

Ebenfalls werden genetische Algorithmen auf die Entwurfsraumanalyse von fehlertoleranten Systemen von Schott [Sch95] angewendet. Die frühe Arbeit definiert im Bereich der multi-kriteriellen Optimierung eine *Spacing*-Metrik die eine Verteilung von nicht-dominierten Individuen auf einer Pareto-Front bewirkt. Dies garantiert eine hohe Abdeckung der Qualitätsattribute und lässt eine Abgrenzung der Individuen zu.

In dieser Arbeit werden die Ergebnisse einer meta-heuristischen Exploration des variantenreichen Problemraums in dem verfolgten Ansatz aufgegriffen. In dieser Vorselektion findet ebenfalls eine qualitative Abgrenzung der Individuen (hier: Konfigurationen) im multi-kriteriellen Raum statt. Die Metrik von Schott quantifiziert die Intensität der qualitativen Abgrenzung Pareto-effizienter Konfigurationen. In dem verfolgten Ansatz werden Konfigurationen anhand ihrer Ressourcenverwendung strukturell abgegrenzt, qualitative Aspekte dienen ausschließlich zur Auflösung von nichtdeterministischen

Entscheidungen im Graphen. Die Anwendung der Spacing-Metrik kann eine präzise Parametrisierung der qualitativen Streuung zwischen Konfigurationsalternativen bezwecken. Dies würde dem Anwender eine zusätzliche Möglichkeit geben die Vorselektion von Konfigurationen hinsichtlich der qualitativen Stabilität von Rekonfigurationen zu beeinflussen.

7.4. Distanzmaße im Lösungsraum

Eine Abgrenzung von Varianten im Lösungsraum erfolgt generell durch eine Erhebung und Untersuchung der Nachbarschaften von Konfigurationen.

Garlan et al. [GBSC09] führen in ihrem Grundlagenwerk zur Architekturevolution Evolutionspfade ein, die eine Restrukturierung von Altlast-Architekturen zu neuen Zielarchitekturen schrittweise begleiten. Wiederholt auftretende Muster werden als Evolutionsstile beschrieben. Auf dieser Grundlage haben Barnes et al. [BPG13] Mechanismen entwickelt, die eine automatisierte Ableitung von Evolutionspfaden zulassen. Auf Grundlage einer spezifizierten Zielarchitektur, durchläuft ein suchbasierter Rekonfigurationsprozess mögliche Pfade zwischen der aktuellen Architektur und der Zielarchitektur. Anschließend wird ein optimaler Evolutionspfad über eine Menge von transienten Architekturen erhoben, siehe Abbildung 7.3.

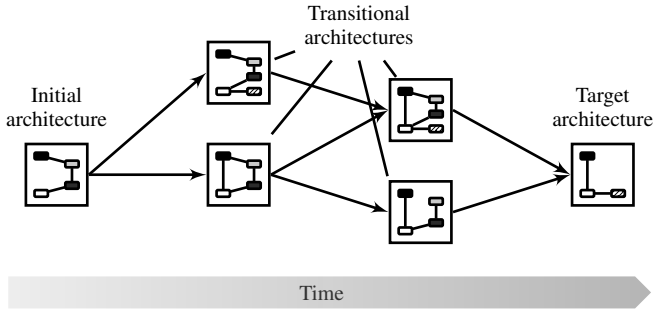


Abbildung 7.3.: Evolutionspfad zwischen Systemarchitekturen (Quelle: [BPG13])

Jeder Knoten beschreibt eine Systemarchitektur, Kanten beschreiben mögliche evolutionäre Transitionen. Dabei wird durch den Algorithmus der Pfad

gewählt, der die Evolutionsziele bestmöglich erzielt. Diese Ziele werden charakterisiert durch Kosten und Dauer der Rekonfiguration und beschränkt durch generelle Vorgaben aus den Evolutionsstilen. Die Erhebung eines optimalen Pfades wird als Planungsproblem in der Sprache PDDL[4] [FL03] aufgefasst und analysiert.

Sowohl der Ansatz von Garlan et al., als auch der von Barnes et al. können zur Erhebung von optimierten Pfaden im Rekonfigurationsraum genutzt werden. Diesbezüglich liegt eine Nähe zu dieser Arbeit vor. In Abgrenzung zur Evolutionspfaden, ist der Ansatz in dieser Arbeit darauf ausgerichtet, eine möglich hohe qualitative Stabilität während einer Rekonfiguration zu gewährleisten. Die Kosten einer Rekonfiguration werden bereits bei der Konstruktion des deterministischen Architekturrelationsgraphens berücksichtigt und beeinflussen die Verknüpfung aller Architekturen. Weiterhin gibt es keine manuelle Vorgabe einer gewünschten Zielarchitektur. Die Suche dieser Alternativen wird durch die Bedingungen und Prioritäten an den Kanten im Graphen gesteuert und führt zu einem Rekonfigurationspfad. Die Idee der Evolutionspfade kann die vorliegende Arbeit bezüglich einer Beurteilung der Einhaltung von Evolutionsstilen ergänzen.

In eine ähnliche Richtung schlagen Kang et al. [KG14] Evolutionspläne zur deterministischen oder adaptiven Anpassung von Softwarearchitekturen vor. In Bezug auf die Beschreibung einer Rekonfiguration, spezifizieren zwei Ausprägungen von Plänen die geplante Evolution eines Systems auf Architekturebene. Deterministische Pläne geben eine sequenzielle Strategie vor, die beschreibt welche statischen Architekturwechsel im Laufe der Evolution stattfinden. Adaptive Pläne, siehe Abbildung 7.4, werden ebenfalls vordefiniert, passen sich jedoch zur Laufzeit an die Umgebungsbedingungen an. Die Knoten in der Abbildung repräsentieren Architekturentwürfe (AD, Englisch: Architecture Designs); die Kanten beschreiben die Übergänge zwischen Architekturen. Jede eingehende Kante in einem adaptiven Plan wird mit einem Unsicherheitsfaktor (P_r) ergänzt. Dieser Faktor priorisiert welche Architektur während einer Rekonfiguration zu wählen ist. Sind diese Werte nicht konstant (vergleiche ausgehende Kante an $AD_{<0>}$) auf Basis von beispielsweise Expertenwissen zur Entwurfszeit festgelegt, werden Variablen zur Laufzeit gebunden um eine kontextabhängige Priorisierung (vergleiche ausgehende Kanten an $AD_{<0,1>}$ und $AD_{<0,2>}$) vorzunehmen. Jede Architektur

[4] Planning Domain Definition Language

wird wirtschaftlich nach Nutzwert und Kosten bewertet. Im Rekonfigurationprozess wird der Pfad mit dem maximalen Wert über alle beteiligten Architekturen gebildet.

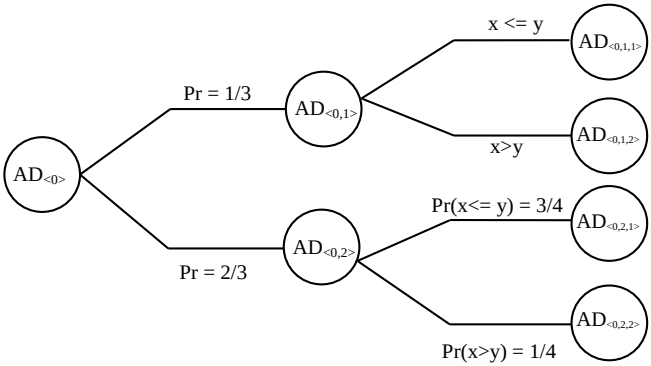


Abbildung 7.4.: Endlicher adaptiver Evolutionsplan (Quelle: [KG14])

In Bezug auf diese Arbeit, entsprechen deterministische Pfade zunächst der Menge aller gültigen Rekonfigurationspfade durch einen deterministischen Architekturrelationsgraphen, solange die Ressourcenplattform vollständig verfügbar ist. Werden jedoch Ressourcenfehler berücksichtigt, verändert sich die Umgebung und Rekonfigurationsentscheidungen werden beeinflusst. Somit wären Rekonfigurationspfade in Form von adaptiven Evolutionsplänen beschreibbar. Dabei sind sowohl qualitative Priorisierung als auch Unsicherheitsfaktoren zu kodieren. Strukturelle Differenzen zwischen Architekturen sind in dem Modell nicht analysierbar abbildbar. Entsprechend resultiert zwar ein deterministischer, jedoch nicht-minimierter Graph. Dies beeinträchtigt die Laufzeit der Exploration und die Nachvollziehbarkeit von Rekonfigurationsbegründungen massiv.

Im Bereich der Service-orientierten Architekturen (SoA) beschreiben Shrivastava et al. [SS13] qualitätsorientierte Rekonfigurationen zur Wiederherstellung von Dienstprozessen. Im Zuge der Dienstkomposition (Englisch: Service Composition) werden alternative Dienste gesucht, die einen ausgefallenen Dienst mit möglichst gleichbleibenden Eigenschaften an die Dienstgüte – Antwortzeit, Verfügbarkeit und Dienstkosten – ersetzen. Zur fehlertoleran-

ten Komposition von Diensten innerhalb von Klassen werden gerichtete azyklische Graphen zur Entwurfszeit festgelegt. Kommt es im Betrieb zu einem Ausfall, wird entweder ab der Position des aktuellen Dienstes in dem Graphen ein alternativer Weg gesucht, oder der Prozess wird neu gestartet und der fehlerhafte Weg umgangen. Die Abbildung 7.5 zeigt einen exemplarischen Graphen mit zwei möglichen alternativen Pfaden.

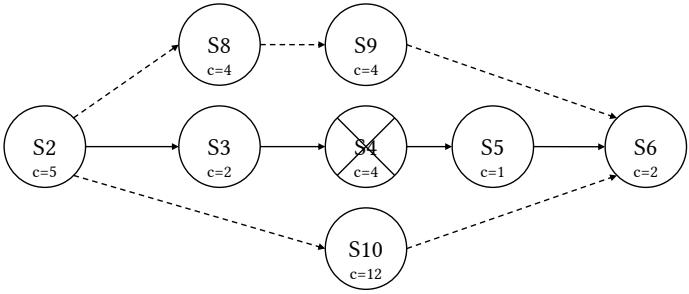


Abbildung 7.5.: Graph einer fehlerhaften Dienstklasse (Quelle: [SS13])

In dem Modell beschreiben die Knoten die Dienste und die Kanten definieren die Reihenfolge der Komposition des Dienstprozesses. Die unterschiedlichen Pfade im Graphen werden durch Budgetgrenzen für die Kosten der Prozesserweiterung beeinflusst. In dem gezeigten Graphen ist eine Anwahl der zwei gestrichelten Pfade möglich, die Kostendifferenzen der alternativen Wege liegen bei 1 beziehungsweise 5 Kosteneinheiten gegenüber des fehlerhaften Pfads. Mit steigender Kompositionsgröße, wächst der Aufwand zur Bestimmung einer fehlerhaften Region, das heißt, die Quantisierung der Auswirkungen eines Knotenausfalls. In einem ähnlichen Ansatz nutzen Canfora et al. [CDPEV08] generische Algorithmen zur Identifikation optimaler Pfade zur Dienstkomposition.

Die Abgrenzung dem hier gewähltem graphbasierten Rekonfigurationsansatz liegt darin, dass Shrivastava et al. ausschließlich die Fehlerbehandlung in ihrem Modell adressieren. Grundsätzlich handelt es sich bei der Exploration um eine Flussoptimierung in einem Netz, die zur Ad-hoc-Fehlerbehandlung zur Laufzeit vorberechnet wird. Die Stärke des Modells liegt in der Identifikation von Regionen, die von einem Fehler betroffen sind. In dem in dieser

Arbeit vorgestellten Ansatz werden fehlerhafte Konfigurationen unmittelbar übersprungen und die Suche fortgesetzt. Ein vorausschauendes Umgehen von Subgraphen, die durch einen Fehler völlig invalide sind, findet indes nicht statt. Weiterhin erfolgt nach Shrivastava et al. eine unmittelbare Abgrenzung der Dienste zueinander ausschließlich in anwenderdefinierten Pfade; eine automatisierte Ableitung der Pfade leistet der Ansatz nicht. Die Modelle in beiden verwandten Ansätzen liefern keine transparenten Begründungen für Rekonfigurationsentscheidungen und qualitative Degradierungen und adressieren keine Redundanzaspekte in eingebetteten Systemen.

7.5. Entscheidungsunterstützung in der Auswahl einer Konfiguration

Die Entscheidungsunterstützung in diesem Ansatz basiert vorrangig auf der Identifikation einer adäquaten Austauschkonfiguration sowie einer nachvollziehbaren Begründung zu dieser Auswahl.

Frey et al. [FFH13] liefern einen Ansatz zur Identifikation von Rekonfigurationsoptionen für Cloud-basierte Systeme. Die Autoren leiten zur Entwurfszeit simulationsbasiert Regeln für systematische Deployment-Änderungen eines Systems ab. Diese Regel werden in Reaktion auf Änderungen des Betriebsumfelds ausgeführt, zum Beispiel Systemüberlasten, um SLA-Verletzung (Englisch: Service-Level-Agreement) zu vermeiden. Ähnlich zu diesem Ansatz adaptieren Jung et al. [JJH+08] virtuelle Maschinen mit Taktiken (Englisch: Policies), die bereits zur Entwurfszeit erhoben werden. Zur Aufstellung der Taktiken nutzen die Autoren maschinelles Lernen auf Entscheidungsbäumen. Das Verfahren wird dabei mit möglichen Systemkonfigurationen trainiert. Zur Laufzeit entscheidet ein Migrationsmechanismus autark, ob eine Änderung der laufenden Konfiguration notwendig ist um eine SLA-Verletzung zu vermeiden.

Beide Ansätze leiten den Anwender in der Identifikation einer betriebsoptimalen Konfiguration des Systems. Dies ähnelt dem Ansatz in dieser Arbeit. Dennoch wird der Rekonfigurationsraum nicht explizit nach qualitativen Einbrüchen durch Ausfälle für potenziell instabile Systeme untersucht. Eine Abwägung mehrere Entwurfsalternativen liegt weiterhin auch nicht vor.

Stattdessen werden Transformationen der aktuellen Systemkonfiguration in eine einzelne neue Konfiguration bereitgestellt. Der Ansatz dieser Arbeit liefert eine Übersicht über kosteneffizient erreichbare Folgekonfigurationen und begründet die Auswahl einer adäquaten Konfiguration durch eine Reihe von qualitativen Kennzahlen aus dem Alternativenvergleich.

Einen hardwarebezogenen Ansatz stellen Malek et al. [MMMR12] im Kontext des Selbstmanagements eingebetteter Systeme [KM07] bereit. Unter Betrachtung von Zielkonflikten zwischen Dienstgüteattributen definieren die Autoren ein Modell zur Bestimmung einer adäquaten Deployment-Architektur für verteilte Software-/Hardwaresysteme. Neben der kontinuierlichen Überwachung der Dienstgüte wird die Anpassung der Architektur durch optimierte Re-Deployment-Operationen ausgeführt. Die Autoren stellen weiterhin ein Framework zur Simulation der Deployments bereit.

Der beschriebene Ansatz ist bezüglich der Abgrenzung von Konfigurationen über eine Nutzwertbildung verwandt zu dieser Arbeit. Dabei wird jedoch keine Priorisierung der Alternativen angeboten. Somit liegt keine Möglichkeit einer vollwertigen Entscheidungsunterstützung zur Laufzeit vor. Weiterhin werden die Rekonfigurationskosten in der Auswahl einer Konfiguration nicht explizit betrachtet. Die Deployment-Operationen werden zwar innerhalb von Simulationen optimiert, jedoch fehlt eine ausführliche Begründung der Anpassungen. Die Transparenz der Wartungsaktivitäten ist notwendig um die Akzeptanz der Entscheidungsträger zu sichern.

7.6. Identifikation und Dokumentation von Aktivitäten im Wartungsprozess

Die Identifikation und Einordnung möglicher Konfigurationsalternativen und die transparente Aufbereitung der Beziehungen bilden einen Mittelpunkt des vorgestellten Ansatzes. In der Literatur werden mögliche Anpassungen in einem System auch unter dem Begriff Flexibilität gegenüber Änderungsanfragen (Englisch: Change Requests) zusammengefasst.

Naab et al. [NS12] beschreiben die systematische Konstruktion und Nutzung der Flexibilität im gesamten Lebenszyklus eines Softwaresystems. Die Autoren schlagen vor, Flexibilität als Qualitätsattribut im Entwicklungsprozess

zu manifestieren und diesbezügliches Wissen aus der Entwurfszeit im Betrieb aufzugreifen. Der Anwender konstruiert zur Entwurfszeit zusätzliche Modellsichten, die Auswirkungen von Modelländerungen beschreiben. Der Detaillierungsgrad der Sichten erlaubt einerseits eine Beurteilung der Flexibilität im Modell, andererseits wird eine geführte Rekonfiguration zur Laufzeit vorbereitet. Steht im Betrieb des Systems eine Änderungsanfrage aus, wird das Wartungspersonal unter Verwendung des annotierten zusätzlichem Entwurfswissens szenarienbasiert durch die notwendigen Modellanpassungen in Form von Arbeitsplänen geleitet.

In Abgrenzung zu dem Ansatz der vorliegenden Arbeit, werden zwar zusätzliche Schnittstellen zur Erweiterung der Software vorbereitet und umfassend erläutert, allerdings werden keine Konfigurationsalternativen ausgeprägt. So wird das Wartungspersonal zwar durch den Problemraum geleitet, konkrete Vorschläge aus dem Lösungsraum möglicher Konfigurationen werden nicht angeboten. Der Ansatz kann ergänzend eingesetzt werden, um Aktionen zu spezifizieren, die bei dem Übergang von Konfigurationen ausgeführt werden müssen. Dadurch kann die Flexibilität als zusätzliches Qualitätsattribut in die Priorisierung von Konfigurationen aufgenommen werden.

In Erweiterung der Arbeit von Naab et al., sind Wartungsaktivitäten auch immer an organisatorische Prozess gekoppelt. Insbesondere sind die Entscheidungsträger über die Folgen umfassender Rekonfigurationen zu informieren. Im Zuge einer quantitativen Architekturevaluation hinsichtlich Wartbarkeit, analysieren Rostami et al. [RSHR15] die Auswirkungen von Änderungen. Durch die Annotation von zusätzlichem technischen (zum Beispiel Deployment-Informationen) als auch organisatorischen Artfakten (zum Beispiel Personaleinsatz) entsteht eine reichhaltige Wissensbasis zur Entwurfszeit. Hieraus lassen sich neben Entwicklungskosten vor allem organisatorische Wartungskosten propagieren. Dies unterstützt das Wartungspersonal in der Kostenbestimmung von Änderungsanfragen und deren Priorisierung. Die Autoren ermöglichen dadurch einerseits eine transparente Beurteilung der Wartbarkeit von Änderungsanfragen, andererseits lassen sich für Wartungen konkrete Arbeitspläne für das Wartungspersonal ableiten.

Der Ansatz unterstützt die Auswahl und Begründung von Wartungsentscheidungen, ausgelöst durch Änderungsanfragen. Auf den verfolgten Ansatz übertragen, würde jede weitere defekte Ressource eine Änderungsanfrage darstellen. Im Anschluss stünden dem Wartungspersonal und den Ent-

scheidungsträgern sämtliche Informationen zur Behebung des Fehlers zur Verfügung. Dies bedarf allerdings einer Vorbereitung sämtlicher Rekonfigurationsszenarien zur Entwurfszeit. Unter Annahme einer variablen Ressourcenplattform ist dies mit einem hohen manuellen Modellierungsaufwand verbunden. Durch eine Integration beider Ansätze könnten die Auswirkungen einer Rekonfigration auf organisatorischer Ebene beschrieben werden. Dies kann in der Domäne der Raumfahrtsysteme vor allem der starken Verteilung von Entwicklergruppen und deren beschränkten Verfügbarkeit entgegenwirken.

8. Schlussfolgerungen und Ausblick

Eine zusammenfassende und kritische Betrachtung der erzielten Ergebnisse sowie ein Ausblick auf mögliche Anschlussarbeiten bilden den Abschluss dieser Arbeit.

8.1. Résumé

Der vorgestellte Ansatz greift qualitative Kennzahlen über die Ressourcen und deren redundanten Kombinationen zur Entwurfszeit auf und manifestiert dieses Wissen in ein Entscheidungsmodell. Der Architekturrelationsgraph (*ARG*) beschreibt dabei einen minimierten zusammenhängenden Entwurfsraum gültiger Konfigurationen. Die Konfigurationen werden als Knoten in absteigender Abhängigkeit zu den eingesetzten Ressourcen sortiert und nach Rekonfigurationsaufwänden abgegrenzt. Dabei erfolgt eine Minimierung des *ARG*, unter Berücksichtigung der Rekonfigurationskosten, die durch strukturelle Differenzen zwischen Konfigurationen mit unterschiedlichem Ressourceneinsatz auftreten. Strukturelle Konfigurationsänderungen geben die Knotenreihenfolge im *ARG* vor. Konkret ausfallgefährdete Ressourcen dienen als Kantenbedingungen, die im Fehlerfall wahr werden. Betriebsmodi spezifizieren Normalisierungen und Gewichtungen der Qualitätsdimensionen, sowie qualitative Grenzwerte. Über die resultierenden prognostizierten Nutzwerte möglicher Alternativkonfigurationen werden Kanten über die prognostizierten Nutzwerte möglicher Alternativkonfigurationen zueinander priorisiert. Hieraus lassen sich Konfigurationsoptionen eindeutig in qualitativen Degradationen abgrenzen und in einem gewichteten deterministischen Architekturrelationsgraphen (*DARG*) pro Szenario ablegen. Tritt zur Laufzeit des Systems ein Fehler auf, erfolgt die Suche im

minimierten Raum der betriebsoptimalen Alternativen. Dies verringert die Suchkomplexität und fördert eine effiziente und transparente Durchführung des Wartungsprozesses, bei minimalen Interaktionen mit den Entscheidungsträgern. Basierend auf der theoretischen Grundlage wird ein prototypisches Werkzeug im Palladio-Umfeld zur Modellierung, Entwurfsraumpriorisierung, Fehlerinjektion, Auswirkungsanalyse und Dokumentation bereitgestellt. Als Anwendungsszenario wurde das Lageregelungssystem des Mikrosatellitens OOV TET-1 in werkzeuggestützten Analysen und einer empirischen Studie untersucht. Dabei wurden die folgenden Beobachtungen gemacht.

Neben der Vertiefung des Prozessverständnisses zur Schärfung der Randbedingungen der Anwendungsdomäne, konnte eine generelle Akzeptanz des Ansatzes als Beratungssystem bestätigt werden. Dabei wurde ein höherer Interaktionsgrad als wesentliche Nebenbedingung zum Einsatz herausgearbeitet. Grundsätzlich wurde die Einbettung eines architekturzentrierten Konfigurationsbegriffs begrüßt, wobei vor allem die zusätzliche Transparenz des Systemzustands und der Begründung von Rekonfigurationsvorschlägen für Entscheidungsträger hervorgehoben wurde.

Die Messungen ergaben, dass trotz der grundsätzlich degradationsfreien originären Missionsanforderungen im realen System, eine Vielzahl von qualitativen Abstufungen im Systementwurf prinzipiell möglich sind. Somit konnten auch heterogene Redundanzen zwischen Ressourcen ermittelt und genutzt werden. Unter Einsatz der prototypischen Implementierung des Ansatzes, konnte eine signifikante Reduktion der personellen Aufwände durch die Vermeidung weitreichender Alternativenvergleiche von über 90% gegenüber der ungeführten Suche erreicht werden. Weiterhin wurden die Rekonfigurationskosten durch die Minimierung der Entscheidungsstruktur, bei gleichbleibender Summe der Nutzwerte, reduziert. Die Kommunikationsaufwände wurden mit Bezug auf die Reduktion der übertragenden Daten, durch die Verwendung des zum System synchronisierten Modells des Rekonfigurationsraums, auf wenige Kilobits reduziert. Obwohl empirisch belegt wurde, dass die Latenzen das wesentliche Problem in der Kommunikation darstellen, kann eine Reduktion der Datenmenge den Prozess durch weniger Kontaktaufnahmen unterstützen.

8.1.1. Vorausberechenbarkeit adäquater Rekonfigurationen zur Entwurfszeit

Die Vorausberechenbarkeit adäquater Rekonfigurationen wird vorrangig durch die Unschärfe von Prognosedaten und der Variabilität des Systems limitiert.

Die möglichen Datenquellen für qualitative Eigenschaften von Ressourcen beschränken sich auf Datenblätter und historische Quellen. Zusätzlich sind Ausführungskontexte auf wenige Konstanten zu reduzieren, um eine effiziente Analyse zu ermöglichen, die unabhängig von einzelnen Betriebsbedingungen ist. Auf Basis dieser Daten ist eine Prognose des Rekonfigurationsraums möglich. Die Auswahl einer konkreten alternativen Konfiguration ist aber nur dann aussagekräftig, wenn die qualitativen Unterschiede in soweit signifikant sind, dass eine Fehlentscheidung durch Prognosetoleranzen auszuschließen ist. In diesem Sinne kann der Ansatz zuverlässig Subgraphen für mögliche Konfigurationen identifizieren, zeigt aber Grenzen bei der exakten Auswahl einer einzelnen Alternativkonfiguration auf. Dies führt dazu, dass die vorliegende Fassung als Beratungssystem im Wartungsprozess dienen kann, jedoch nicht als Werkzeug zur autonomen Rekonfiguration.

Weiterhin ist die Anwendung des Ansatzes nur lukrativ, wenn das System einen großen Raum an Variationen zulässt. Erst mit einer Vielzahl und Heterogenität valider Redundanzgruppierungen werden Rekonfigurationen so komplex, dass manuelle Explorationen unwirtschaftlich sind.

Dieses Problem wächst an, wenn die Kostenfunktion für Rekonfigurationen zusätzlich eine laufzeitdynamische Komponente enthält. Dies ist zu dann berücksichtigen, wenn eine Rekonfiguration den Aktivierungszustand kalt-redundanter Ressourcen betrachtet. Eine Vorberechnung ist dabei auch auf Basis der Kombinatorik aller kalt-redundanten Ressourcen möglich, skaliert jedoch nicht. Der vorgestellte Ansatz betrachtet daher die höchstmöglichen Kosten, das heißt dass jede Ressource als heißredundant angenommen wird.

Die domänenspezifische Entwurfsraumexploration erfordert die Definition implementierungsnaher Systemmodelle, Betriebsmodi, Analysedimensionen und zugehöriger Metriken. Die Anwendung des Ansatzes erfordert dabei, dass der Entwicklungsprozess streng modellbasiert ausgerichtet ist und die

Systemarchitektur kontinuierlich konsistent zur produktiven Implementierung gehalten wird. Dabei müssen Ausprägungen der Ressourcenkonstellationen und Varianten der Betriebssoftware auf Architekturebene sichtbar sein. Weiterhin sind Betriebsmodi in Form von Gewichtungen gegenläufiger qualitativer Attribute und zugehörigen Untergrenzen zu definieren. Die Entwicklung der ESA-Initiative EGS-CC, vergleiche Abschnitt 7.1, zeigt eine deutliche Tendenz zur Modernisierung softwareintensiver Systemen im Bodensegment. Weiterhin etabliert die ESA den modellbasierten Datenaustausch im gesamten Lebenszyklus eines Systems,vergleiche *ECSS-E-TM-10-23A* in Abschnitt 7.1. Eine Integration von modellbasierten Techniken in den Wartungsprozess ist somit vielversprechend und findet in Teilen bereits statt.

8.1.2. Bedingungen zur Domänenübertragbarkeit

Der vorgestellte Ansatz ist generisch aufgebaut, stellt aber Voraussetzungen an die Anwendungsdomäne des untersuchten Systems.

Zunächst muss ein Konfigurationsbegriff verfügbar oder zumindest aus den Daten ermittelbar sein. Dies beinhaltet, dass Konfigurationen eindeutig voneinander abgrenzbar sind. Einerseits wird zur Kantenminimierung des Rekonfigurationsraums eine Kostenfunktion benötigt, andererseits eine qualitative Bewertung jeder Konfiguration zur Kantenpriorisierung. Zur wirtschaftlichen Anwendung sollte der Raum eine inhärente Variabilität besitzen und qualitative Degradation zulassen.

Eine Anforderung an die Entwickler ist, dass die Expertise zur frühzeitigen Beurteilung von Konfigurationen vorhanden ist, um Prognosewerte zu beurteilen. Dazu sind beispielsweise Erfahrungen aus Vorgängersystemen aufzugreifen.

Dennoch sollte auch eine Testplattform vorliegen, um die Gültigkeit der Qualitätsprognosen nach aktuellen Betriebsbedingungen zu überprüfen.

Zum Einsatz des Ansatzes zur autonomen Entscheidungsfestlegung ist ein hohes Maß an Akzeptanz für die Autonomie in der Domäne notwendig. Dabei sollte der Ansatz in einem wohldefinierten Rahmen agieren, der den Entwicklern bekannt ist und zu allen Umweltbedingungen kompatibel ist. Eine domänenübergreifende Reproduzierbarkeit der Ergebnisse ist somit

grundsätzlich möglich, ist aber nicht Teil dieser Arbeit. Mögliche Anwendungsgebiete werden jedoch in Aussicht gestellt.

8.2. Ausblick

Die Kriterien in der vorliegenden Arbeit wurden in der Domäne der Raumfahrtsysteme, Teilbereich der Raumflugkörper und Satelliten, validiert. Die wesentliche Motivation lag dabei in der Fernwartung einer nicht-reparablen Ressourcenplattform. Wie in der Einleitung durch das durchgängige Beispiel angedeutet, ist eine Übertragung des Ansatzes auch auf die Automotive-Domäne denkbar. Hier ist die Ressourcenplattform zwar grundsätzlich zugänglich und reparabel, aus logistischen Gründen ist dies aber bei hoher Stückzahl der Fahrzeuge wirtschaftlich nicht tragbar. Der Ansatz sollte aufgrund der sicherheitskritischen Aspekte auch hier als Beratungssystem in der Wartung, zum Beispiel in der Hauptuntersuchung, dienen, um beispielsweise kostengünstige Reparaturen zu unterstützen. Als weitere Domäne mit qualitativen Degradationen wären Produktionsanlagen in der Automatisierungstechnik zu untersuchen. Hier ist in der Regel ein unterbrechnungsfreier Betrieb notwendig. Der Ansatz kann dabei als autonomer Entscheidungsmechanismus dazu dienen, um eine Teilproduktion mit geringer Stückzahl fortzusetzen, wenn beispielsweise primäre Sensorik ausfällt, Rückschlüsse aber noch über sekundäre Sensoren möglich sind. Da in dieser Domäne die Hardwareressourcen grundsätzlich austauschbar sind, kann der Ansatz insbesondere auch dafür abgewandelt werden eine effiziente Umrüstung im Vorfeld bezüglich möglicher Optionen zu planen.

In der vorliegenden Fassung des Ansatzes ist die Erfassung von Werten für die Qualitätsdimensionen auf Basis von Datenblättern beschrieben. Eine Erweiterung des Modells um dynamische Funktionen ist dahin zielführend, dass frühzeitige Simulationen auf dem Teststand in die Prognose nahtlos integriert werden könnten. Dies würde den empirisch erhobenen Bedarf an der Präzisierung der Prognosedaten aufgreifen.

Weiterhin ist eine Anpassung der derzeit linearen Kumulation von Qualitätsdimensionen zu Nutzwerten einer Konfiguration möglich. Unter Berücksichtigung funktionaler Zusammenhänge in Abhängigkeit des Wertespektrums ist das Modell erweiterbar. Dies war jedoch nicht Gegenstand der Arbeit, da

die Erzeugung der Nutzwerte nicht im Fokus stand und die resultierenden Werte der Fallstudie hinreichend verschieden waren. Zur praktischen Anwendung wäre eine solche Modellerweiterung allerdings sinnvoll, um komplexe Zusammenhänge von einzelnen Dimensionen in Differentialgleichungen zu beschreiben.

Eine mögliche Erweiterung wäre die explorative Erhebung des Schwellwerts für Rekonfigurationskosten. In einer weiteren Analyse auf dem vollständigen Rekonfigurationsraum ist dabei zu prüfen, welche Kostenschwelle den durchschnittlichen Graphdurchmesser über alle Betriebsmodi maximiert und somit transiente Konfigurationen in den Rekonfigurationspfaden optimal ausnutzt. Für das Anwendungsszenario in dieser Arbeit wurde der zunächst abgeschätzte Schwellwert nachträglich bewertet, vergleiche Metrik 6.6.4.3. Dabei war bei Absenkungen und Steigerungen keine Verbesserung der zuvor festgelegten Minimierung des *DARG* feststellbar. Diese Annahme kann aber nicht verallgemeinert werden, womit eine Exploration des Schwellwertes vielversprechend ist.

Aktuell werden alternative Konfigurationen auf Grundlage der direkt nachgelagerten Alternativen beurteilt. Die Beurteilung führt kurzfristig zu optimalen Ergebnissen, garantiert aber keine langfristig begründeten Entscheidungen. Entsprechend wäre die Analyse um eine Beurteilung zur ergänzen, die vor Auswahl einer Alternative auch eine Bewertung deren möglicher Nachfolgermengen vornimmt. Hierfür können die Erkenntnisse aus der Erweiterung für variierende Betriebsmodi adaptiert werden. Durch die Beurteilung der Graphzentralität einer jeden Alternative zur Laufzeit, kann deren Bedeutung im *DARG* näher bestimmt werden. Diese Analyse würde das Laufzeitverhalten der Exploration allerdings stark beeinflussen. Der vorliegende Ansatz adressiert hingegen niedrige Laufzeitaufwände.

Ebenfalls wäre eine Erweiterung des Fehlermodells denkbar, welches Abstufungen im Fehlerzustand und Wiederherstellungsfähigkeiten berücksichtigt. Das Datenmodell des Ansatzes unterstützt dies bereits, da ausgefallene Konfigurationen nicht aus dem Graphen entfernt, sondern in der Exploration nur übersprungen werden. Eine Anpassung der Explorationsalgorithmen ist möglich, beeinflusst aber die Laufzeit der Suche. Die Wiederherstellung jeder Konfiguration müsste kontinuierlich geprüft werden. Weiterhin sinkt die Transparenz, da die Knoten im *DARG* mit hoher Frequenz ein- und

ausgeblendet werden. Dies bedeutet für die Dokumentation, dass für jeden Rekonfigurationspfad auch der Gesamtzustand des *DARG* festgehalten werden müsste.

Fazit: Der vorgestellte Ansatz dient zur umfassenden Erhebung, Begründung und Dokumentation von Rekonfigurationsentscheidungen für softwareintensive technische Systeme. Wesentliche Anwendungsvorteile ergeben sich durch die Verlagerung kostenintensiver Untersuchungen des Konfigurationsraums von der Laufzeit in die Entwurfszeit. Dazu werden zum einen während der Entwurfszeit mögliche Konfigurationen ermittelt, entlang qualitativer Kriterien bewertet und kostenorientiert ausgewählt. Durch die Berücksichtigung von Betriebsanforderungen wird eine qualitätsorientierte Priorisierung hin zu einem deterministischen Expertensystem durchgeführt. Zum anderen wird das Wissen aus dem Entwurf als ein zum Produktivsystem synchronisiertes Modell im Betrieb verfügbar gemacht, um auch während der Laufzeit effektive Rekonfigurationsvorschläge effizient zu erheben und zu begründen. Für den langfristigen Betrieb werden vergangene Rekonfigurationen ausführlich dokumentiert. Der Ansatz wurde prototypisch umgesetzt, werkzeuggestützt quantitativ analysiert und in einer empirischen Studie qualitativ auf seine Anwendbarkeit untersucht. Mögliche Ergänzungen des Ansatzes sowie Übertragungen auf andere Anwendungsgebiete erscheinen vielversprechend, wenn manuelle Rekonfigurationen kostenintensiv, der Rekonfigurationsraum adäquat formalisierbar und Rekonfigurationsoptionen frühzeitig bewertbar sind.

A. Unterlagen zur Modellierung

A.1. Freiheitsgrad-Optionen und Ressourcen

Tabelle A.1.: Zuordnung Freiheitsgrad-Optionen zu Ressourcen

DoF Option	Ressourcengruppe	Ressourcenmenge	Min.
MCS 3 Coils	MCS X 3 Coils	MC1x, MC2x	1
MCS 3 Coils	MCS Y 3 Coils	MC1y, MC2y	1
MCS 3 Coils	MCS Z 3 Coils	MC1z, MC2z	1
MCS 6 Coils	MCS X 6 Coils	MC1x, MC2x	2
MCS 6 Coils	MCS Y 6 Coils	MC1y, MC2y	2
MCS 6 Coils	MCS Z 6 Coils	MC1z, MC2z	2
RW Dummy	RW Dummy	$\emptyset$	0
RW 3 Wheels	RW 3 Wheels	RW1, RW2, RW3, RW4	3
RW 4 Wheels	RW 4 Wheels	RW1, RW2, RW3, RW4	4
ASC Dummy	ASC Dummy	$\emptyset$	0
ASC High Res.	ASC CHU High Res.	ASC CHU1, ASC CHU2	1
ASC High Res.	ASC DPU High Res.	ASC DPU1 High Res., ASC DPU2 High Res.	1
ASC Low Res.	ASC CHU Low Res.	ASC CHU1, ASC CHU2	1
ASC Low Res.	ASC DPU Low Res.	ASC DPU1 Low Res., ASC DPU2 Low Res.	1
CSS Dummy	CSS Dummy	$\emptyset$	0
CSS High Res.	CSS Frontheads High Res.	CSS Frontheads1, CSS Frontheads2	1

Wird fortgesetzt…

Zuordnung Freiheitsgrad-Optionen zu Ressourcen (fortgesetzt)

DoF Option	Ressourcengruppe	Ressourcenmenge	Min.
CSS High Res.	CSS Rearheads High Res.	CSS Rearhead1, CSS Rearhead2	1
CSS High Res.	CSS Chipsets High Res.	CSS Chipset1 High Res., CSS Chipset2 High Res	1
CSS Low Res.	CSS Frontheads Low Res.	CSS Frontheads1, CSS Frontheads2	1
CSS Low Res.	CSS Rearheads Low Res.	CSS Rearhead1, CSS Rearhead2	1
CSS Low Res.	CSS Chipsets Low Res.	CSS Chipset1 Low Res., CSS Chipset2 Low Res	1
MFS Dummy	MFS Dummy	∅	0
MFS	MFS	MFS Fluxgate1, MFS Fluxgate2	1
ONS Dummy	ONS Dummy	∅	0
ONS	ONS Receivers	GPS Receiver1, GPS Receiver2	1
ONS	ONS Antennas+LNAs	GPS LNA1+Antenna1, GPS LNA2+Antenna2	1
(Standard)	IMU	IMU1, IMU2	1

A.2. Wertebelegungen für Konfigurationen

Tabelle A.2.: Normalisierte Wertebelegungen für Konfigurationskandidaten

Id	Noise	Lifetime	Pointingtime	Accuracy	Torque	Powerconsumption	Pareto-optimal
4731	0,0667	0,8370	0,0343	0,9677	0,8448	0,0969	
4628	0,1333	0,8370	0,0515	0,8710	0,8448	0,1079	
4500	0,0000	0,9620	0,0000	1,0000	0,8276	0,1648	
4622	0,0667	0,9620	0,0343	0,9677	0,8276	0,1703	
4626	0,0667	0,7989	0,0172	0,9032	0,6724	0,2672	
4513	0,0000	0,8370	0,0000	1,0000	0,8448	0,0915	
4403	0,1333	1,0000	0,0515	0,8710	1,0000	0,0165	
4542	0,0667	1,0000	0,0172	0,9032	1,0000	0,0110	
4424	0,1333	0,7989	0,0515	0,8710	0,6724	0,2727	
4425	0,0000	0,7989	0,0000	1,0000	0,6724	0,2562	
4668	0,0000	1,0000	0,0000	1,0000	1,0000	0,0000	
4453	0,1333	0,9620	0,0515	0,8710	0,8276	0,1813	
4471	0,0667	0,8370	0,0172	0,9032	0,8448	0,1024	
4475	0,0667	1,0000	0,0343	0,9677	1,0000	0,0055	
4346	0,0667	0,9620	0,0172	0,9032	0,8276	0,1758	
4388	0,0667	0,7989	0,0343	0,9677	0,6724	0,2617	
4728	0,3333	0,7283	0,3951	0,6774	0,8448	0,1491	
4623	0,3333	0,8913	0,3951	0,6774	1,0000	0,0577	
4648	0,2667	0,8913	0,3607	0,7097	1,0000	0,0522	
4541	0,2000	0,6902	0,3435	0,8065	0,6724	0,2974	
4421	0,2000	0,8913	0,3435	0,8065	1,0000	0,0412	
4431	0,2667	0,7283	0,3779	0,7742	0,8448	0,1381	
4311	0,2667	0,7283	0,3607	0,7097	0,8448	0,1436	
4559	0,2667	0,8533	0,3779	0,7742	0,8276	0,2115	x

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4686	0,3333	0,6902	0,3951	0,6774	0,6724	0,3139	
4447	0,2667	0,8913	0,3779	0,7742	1,0000	0,0467	
4340	0,2000	0,8533	0,3435	0,8065	0,8276	0,2060	
4341	0,3333	0,8533	0,3951	0,6774	0,8276	0,2225	
4347	0,2000	0,7283	0,3435	0,8065	0,8448	0,1327	
4368	0,2667	0,6902	0,3607	0,7097	0,6724	0,3084	
4701	0,2667	0,6902	0,3779	0,7742	0,6724	0,3029	
4710	0,2667	0,8533	0,3607	0,7097	0,8276	0,2170	
4722	0,2667	0,6739	0,0515	0,7742	0,8448	0,1436	
4602	0,0667	0,7174	0,0209	0,7097	0,6724	0,2796	
4724	0,2000	0,7174	0,0725	0,5806	0,6724	0,2961	
4726	0,1333	0,8804	0,0381	0,6129	0,8276	0,1991	
4734	0,1333	0,8478	0,0515	0,8710	0,3103	0,6756	
4641	0,0667	0,6848	0,0343	0,9677	0,1552	0,7561	
4521	0,2667	0,8370	0,0515	0,7742	1,0000	0,0522	
4402	0,2667	0,7989	0,0515	0,7742	0,8276	0,2170	
4536	0,2000	0,9185	0,0725	0,5806	1,0000	0,0398	
4539	0,0667	0,7554	0,0209	0,7097	0,8448	0,1148	
4423	0,0667	0,6848	0,0172	0,9032	0,1552	0,7616	
4304	0,3333	0,8370	0,0859	0,7419	1,0000	0,0577	x
4667	0,1333	0,8804	0,0553	0,6774	0,8276	0,1936	
4426	0,1333	0,7174	0,0381	0,6129	0,6724	0,2906	
4308	0,1333	0,9185	0,0553	0,6774	1,0000	0,0288	x
4322	0,3333	0,6359	0,0859	0,7419	0,6724	0,3139	
4443	0,0667	0,8478	0,0343	0,9677	0,3103	0,6646	
4313	0,1333	0,6848	0,0515	0,8710	0,1552	0,7671	
4318	0,0667	0,8804	0,0209	0,7097	0,8276	0,1881	
4445	0,2000	0,7554	0,0725	0,5806	0,8448	0,1313	
4583	0,1333	0,7174	0,0553	0,6774	0,6724	0,2851	
4584	0,2000	0,8804	0,0725	0,5806	0,8276	0,2046	

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4579	0,3333	0,7989	0,0859	0,7419	0,8276	0,2225	
4474	0,1333	0,9185	0,0381	0,6129	1,0000	0,0343	
4466	0,1333	0,7554	0,0553	0,6774	0,8448	0,1203	
4588	0,0667	0,8478	0,0172	0,9032	0,3103	0,6702	
4468	0,1333	0,7554	0,0381	0,6129	0,8448	0,1258	
4599	0,2667	0,6359	0,0515	0,7742	0,6724	0,3084	
4492	0,3333	0,6739	0,0859	0,7419	0,8448	0,1491	x
4494	0,0000	0,8478	0,0000	1,0000	0,3103	0,6592	
4379	0,0667	0,9185	0,0209	0,7097	1,0000	0,0233	
4709	0,0000	0,6848	0,0000	1,0000	0,1552	0,7506	
4606	0,3333	0,8206	0,0687	0,8710	0,8276	0,1840	
4607	0,5333	0,5652	0,4294	0,5484	0,8448	0,1903	
4729	0,4000	0,6467	0,4160	0,3871	0,8448	0,1725	
4721	0,4667	0,7283	0,3951	0,5806	1,0000	0,0934	
4601	0,4000	0,6576	0,0859	0,7742	0,6724	0,2865	
4603	0,2000	0,5761	0,3435	0,8065	0,1552	0,7918	
4725	0,2667	0,7717	0,3645	0,5161	0,8276	0,2293	
4605	0,5333	0,6902	0,4294	0,5484	0,8276	0,2637	
4620	0,2667	0,7391	0,3607	0,7097	0,3103	0,7113	
4621	0,3333	0,6467	0,3988	0,4839	0,8448	0,1615	
4625	0,4000	0,6087	0,4160	0,3871	0,6724	0,3373	
4748	0,2667	0,7391	0,3779	0,7742	0,3103	0,7058	
4640	0,2667	0,5761	0,3779	0,7742	0,1552	0,7973	
4518	0,5333	0,7283	0,4294	0,5484	1,0000	0,0989	
4512	0,2667	0,6467	0,3645	0,5161	0,8448	0,1560	
4515	0,4000	0,8098	0,4160	0,3871	1,0000	0,0810	
4409	0,2667	0,6087	0,3645	0,5161	0,6724	0,3208	
4400	0,3333	0,8587	0,0687	0,8710	1,0000	0,0192	
4643	0,4667	0,5652	0,3951	0,5806	0,8448	0,1848	
4407	0,3333	0,5761	0,3951	0,6774	0,1552	0,8083	

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4538	0,4667	0,5272	0,3951	0,5806	0,6724	0,3496	
4430	0,3333	0,6956	0,0687	0,8710	0,8448	0,1107	
4309	0,4000	0,7717	0,4160	0,3871	0,8276	0,2458	
4543	0,3333	0,7391	0,3951	0,6774	0,3103	0,7168	
4664	0,3333	0,6087	0,3988	0,4839	0,6724	0,3263	
4554	0,3333	0,8098	0,3817	0,4194	1,0000	0,0755	
4556	0,4667	0,6902	0,3951	0,5806	0,8276	0,2582	
4567	0,4000	0,8206	0,0859	0,7742	0,8276	0,1950	
4582	0,2000	0,7391	0,3435	0,8065	0,3103	0,7004	
4344	0,2667	0,8098	0,3645	0,5161	1,0000	0,0645	x
4455	0,4000	0,8587	0,0859	0,7742	1,0000	0,0302	
4578	0,3333	0,7717	0,3817	0,4194	0,8276	0,2403	
4470	0,3333	0,7717	0,3988	0,4839	0,8276	0,2348	
4476	0,3333	0,6087	0,3817	0,4194	0,6724	0,3318	
4477	0,5333	0,5272	0,4294	0,5484	0,6724	0,3551	
4498	0,2667	0,5761	0,3607	0,7097	0,1552	0,8028	
4705	0,3333	0,8098	0,3988	0,4839	1,0000	0,0700	
4706	0,4000	0,6956	0,0859	0,7742	0,8448	0,1217	
4389	0,3333	0,6467	0,3817	0,4194	0,8448	0,1670	
4700	0,3333	0,6576	0,0687	0,8710	0,6724	0,2755	
4619	0,3333	0,7554	0,0725	0,4839	1,0000	0,0755	
4627	0,6000	0,7500	0,4294	0,5806	1,0000	0,0714	
4646	0,4000	0,5544	0,1068	0,4516	0,6724	0,3373	
4647	0,6000	0,5489	0,4294	0,5806	0,6724	0,3277	
4661	0,6000	0,5870	0,4294	0,5806	0,8448	0,1629	
4420	0,3333	0,6848	0,0859	0,7419	0,3103	0,7168	
4663	0,5333	0,5489	0,4122	0,6774	0,6724	0,3167	
4310	0,2000	0,6033	0,0725	0,5806	0,1552	0,7904	
4303	0,6000	0,7120	0,4294	0,5806	0,8276	0,2362	
4306	0,4000	0,5924	0,1068	0,4516	0,8448	0,1725	

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4307	0,0667	0,7663	0,0209	0,7097	0,3103	0,6825	
4561	0,1333	0,6033	0,0553	0,6774	0,1552	0,7795	
4683	0,4000	0,7554	0,1068	0,4516	1,0000	0,0810	
4442	0,5333	0,7120	0,4122	0,6774	0,8276	0,2252	
4685	0,5333	0,7500	0,4122	0,6774	1,0000	0,0604	x
4312	0,0667	0,6033	0,0209	0,7097	0,1552	0,7740	
4452	0,3333	0,5924	0,0725	0,4839	0,8448	0,1670	
4687	0,2667	0,5217	0,0515	0,7742	0,1552	0,8028	
4446	0,3333	0,5544	0,0725	0,4839	0,6724	0,3318	
4688	0,3333	0,7174	0,0725	0,4839	0,8276	0,2403	
4580	0,1333	0,7663	0,0553	0,6774	0,3103	0,6880	
4581	0,1333	0,6033	0,0381	0,6129	0,1552	0,7850	
4469	0,1333	0,7663	0,0381	0,6129	0,3103	0,6935	x
4360	0,3333	0,5217	0,0859	0,7419	0,1552	0,8083	x
4361	0,4000	0,7174	0,1068	0,4516	0,8276	0,2458	
4365	0,5333	0,5870	0,4122	0,6774	0,8448	0,1519	
4488	0,2667	0,6848	0,0515	0,7742	0,3103	0,7113	
4703	0,2000	0,7663	0,0725	0,5806	0,3103	0,6990	
4608	0,3333	0,5435	0,0687	0,8710	0,1552	0,7698	
4604	0,5333	0,5761	0,4294	0,5484	0,3103	0,7580	
4730	0,5333	0,6087	0,4160	0,2903	0,8276	0,2815	
4751	0,5333	0,4837	0,4160	0,2903	0,8448	0,2082	
4624	0,6000	0,6467	0,4504	0,2581	1,0000	0,1222	
4746	0,5333	0,4456	0,4160	0,2903	0,6724	0,3730	
4520	0,2667	0,4946	0,3645	0,5161	0,1552	0,8152	
4408	0,2667	0,6576	0,3645	0,5161	0,3103	0,7237	
4401	0,4000	0,7391	0,0897	0,5806	0,8276	0,2074	
4645	0,6000	0,4837	0,4504	0,2581	0,8448	0,2137	
4662	0,4667	0,5761	0,3951	0,5806	0,3103	0,7525	
4534	0,6000	0,4456	0,4504	0,2581	0,6724	0,3785	

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4537	0,4667	0,6141	0,1068	0,4839	0,8448	0,1450	
4659	0,4667	0,5761	0,1068	0,4839	0,6724	0,3098	
4305	0,3333	0,7065	0,0687	0,8710	0,3103	0,6784	
4560	0,4000	0,6576	0,4160	0,3871	0,3103	0,7402	
4563	0,4000	0,7772	0,0897	0,5806	1,0000	0,0426	
4684	0,3333	0,4946	0,3817	0,4194	0,1552	0,8262	
4555	0,3333	0,6576	0,3988	0,4839	0,3103	0,7292	
4314	0,5333	0,6467	0,4160	0,2903	1,0000	0,1167	
4319	0,4000	0,5435	0,0859	0,7742	0,1552	0,7808	
4450	0,4667	0,7391	0,1068	0,4839	0,8276	0,2183	
4585	0,3333	0,6576	0,3817	0,4194	0,3103	0,7347	
4334	0,6000	0,6576	0,1202	0,6452	0,8276	0,2362	
4336	0,6000	0,5326	0,1202	0,6452	0,8448	0,1629	
4699	0,4000	0,4946	0,4160	0,3871	0,1552	0,8316	
4337	0,4000	0,7065	0,0859	0,7742	0,3103	0,6894	
4338	0,4667	0,7772	0,1068	0,4839	1,0000	0,0536	x
4362	0,4000	0,5761	0,0897	0,5806	0,6724	0,2988	
4363	0,6000	0,4946	0,1202	0,6452	0,6724	0,3277	
4359	0,6000	0,6087	0,4504	0,2581	0,8276	0,2870	
4495	0,4667	0,4130	0,3951	0,5806	0,1552	0,8440	
4381	0,4000	0,6141	0,0897	0,5806	0,8448	0,1340	
4382	0,6000	0,6956	0,1202	0,6452	1,0000	0,0714	x
4385	0,3333	0,4946	0,3988	0,4839	0,1552	0,8206	
4387	0,5333	0,4130	0,4294	0,5484	0,1552	0,8495	
4750	0,6000	0,4348	0,4294	0,5806	0,1552	0,8220	
4745	0,6000	0,6304	0,4332	0,3871	0,8276	0,2486	
4639	0,8000	0,5489	0,4638	0,4516	0,8276	0,2774	
4519	0,6667	0,6304	0,4504	0,2903	0,8276	0,2595	
4517	0,4000	0,4402	0,1068	0,4516	0,1552	0,8316	
4644	0,8000	0,3859	0,4638	0,4516	0,6724	0,3689	x

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4406	0,6000	0,4674	0,4332	0,3871	0,6724	0,3400	
4422	0,6667	0,6685	0,4504	0,2903	1,0000	0,0948	
4681	0,6667	0,5054	0,4504	0,2903	0,8448	0,1862	
4564	0,5333	0,4348	0,4122	0,6774	0,1552	0,8110	
4316	0,6000	0,5054	0,4332	0,3871	0,8448	0,1752	x
4323	0,3333	0,4402	0,0725	0,4839	0,1552	0,8262	x
4343	0,6000	0,6685	0,4332	0,3871	1,0000	0,0838	
4587	0,8000	0,4239	0,4638	0,4516	0,8448	0,2041	x
4364	0,6000	0,5978	0,4294	0,5806	0,3103	0,7306	
4496	0,6667	0,4674	0,4504	0,2903	0,6724	0,3510	
4489	0,8000	0,5870	0,4638	0,4516	1,0000	0,1126	
4384	0,3333	0,6033	0,0725	0,4839	0,3103	0,7347	
4380	0,5333	0,5978	0,4122	0,6774	0,3103	0,7196	
4707	0,4000	0,6033	0,1068	0,4516	0,3103	0,7402	
4600	0,6667	0,4130	0,1412	0,3548	0,6724	0,3510	
4511	0,6000	0,3315	0,4504	0,2581	0,1552	0,8728	
4404	0,6667	0,4511	0,1412	0,3548	0,8448	0,1862	
4405	0,4000	0,4620	0,0897	0,5806	0,1552	0,7932	
4660	0,5333	0,3315	0,4160	0,2903	0,1552	0,8673	x
4532	0,4667	0,6250	0,1068	0,4839	0,3103	0,7127	
4665	0,4667	0,4620	0,1068	0,4839	0,1552	0,8042	
4682	0,6667	0,5761	0,1412	0,3548	0,8276	0,2595	
4679	0,6000	0,4946	0,4504	0,2581	0,3103	0,7814	
4317	0,4000	0,6250	0,0897	0,5806	0,3103	0,7017	
4444	0,6000	0,5435	0,1202	0,6452	0,3103	0,7306	
4358	0,6000	0,3804	0,1202	0,6452	0,1552	0,8220	
4490	0,5333	0,4946	0,4160	0,2903	0,3103	0,7759	
4383	0,6667	0,6141	0,1412	0,3548	1,0000	0,0948	
4642	0,6667	0,3533	0,4504	0,2903	0,1552	0,8454	
4302	0,8000	0,2717	0,4638	0,4516	0,1552	0,8632	x

Wird fortgesetzt…

Normalisierte Wertebelegungen für Konfigurationskandidaten (fortgesetzt)

Id	Noise	Lifetime	Pointingt.	Accuracy	Torque	Powercon.	Pareto-opt.
4428	0,6667	0,5163	0,4504	0,2903	0,3103	0,7539	
4680	0,6000	0,5163	0,4332	0,3871	0,3103	0,7429	
4497	0,8000	0,4348	0,4638	0,4516	0,3103	0,7718	
4704	0,6000	0,3533	0,4332	0,3871	0,1552	0,8344	
4749	0,6667	0,4620	0,1412	0,3548	0,3103	0,7539	
4540	0,6667	0,2989	0,1412	0,3548	0,1552	0,8454	

B. Daten der werkzeug-gestützten Messung

B.1. Tabellarische Rohdaten

Tabelle B.1.: Ressourcendifferenzmengen der Pfadkonfigurationen des Betriebsmodus N7

Quelle	Ziel	Abzüge	Hinzunahmen
4567	4338	GPS Receiver2	∅
4338	4401	MFS Fluxgate1; ASC DPU2 Low Res.; ASC DPU1 Low Res.	ASC DPU2 High Res.
4401	4381	MC1x	∅
4381	4362	RW2	∅

Tabelle B.2.: Ressourcendifferenzmengen der Pfadkonfigurationen des Betriebsmodus N15

Quelle	Ziel	Abzüge	Hinzunahmen
4726	4450	CSS FrontHead1; CSS FrontHead2; CSS Chipset2 High Res.	∅
4450	4401	ASC DPU1 Low Res.	ASC DPU2 High Res.
4401	4316	MFS Fluxgate2; MC1x	∅
4316	4406	RW2	∅

Tabelle B.3.: Pfadbasierte Kennzahlen aus acht Minimierungsstufen (Grenzen) für alle Betriebsmodi

Grenze	Mission	Knoten	Kanten	Durchmesser
4	N1	124	262	4
4	N7	104	274	3
4	N15	110	206	3
4		**112**	**247**	**3.3**
8	N1	124	1546	6
8	N7	104	1262	8
8	N15	110	1236	6
8		**112**	**1348**	**6.7**
12	N1	124	3516	4
12	N7	104	2742	4
12	N15	110	2926	4
12		**112**	**3061**	**4.0**
16	N1	124	6440	4
16	N7	104	4972	4
16	N15	110	5300	4
16		**112**	**5570**	**4.0**
20	N1	124	10154	3
20	N7	104	7502	3
20	N15	110	8228	3
20		**112**	**8628**	**3.0**
24	N1	124	12838	3
24	N7	104	9278	3
24	N15	110	10298	3
24		**112**	**10804**	**3.0**
28	N1	124	14532	3
28	N7	104	10342	3
28	N15	110	11510	3
28		**112**	**12128**	**3.0**
∞	N1	124	15252	2
∞	N7	104	10712	2
∞	N15	110	11990	2
∞		**112**	**12651**	**2.0**

Tabelle B.4.: Optimale Fehlersequenzen für alle Betriebsmodi

Schritt	N1	N5	N7
1	GPS LNA1+Antenna1	GPS LNA1+Antenna1	GPS LNA1+Antenna1
2	GPS Receiver1	MC1y	GPS Receiver1 / **CSS Chipset2 High Res.**
3	GPS LNA2+Antenna2	GPS LNA2+Antenna2	GPS LNA2+Antenna2
4	MFS Fluxgate1 / **MC1y**	MFS Fluxgate1	MFS Fluxgate1
5	CSS RearHead2	CSS RearHead2	CSS RearHead2
6	ASC DPU1 High Res.	ASC DPU1 High Res.	ASC DPU1 High Res.
7	ASC DPU2 Low Res.	ASC DPU2 Low Res.	ASC DPU2 Low Res.
8	CSS Chipset1 High Res.	CSS Chipset1 High Res.	CSS Chipset1 High Res.
9	CSS RearHead1	CSS RearHead1	CSS RearHead1
10	CSS FrontHead2	CSS FrontHead2	CSS FrontHead2
11	CSS Chipset2 Low Res.	CSS Chipset2 Low Res.	CSS Chipset2 Low Res.
12	ASC DPU1 Low Res.	ASC DPU1 Low Res.	ASC DPU1 Low Res.
13	MC1x	MC1x	MC1x
14	RW2	RW2	RW2
15	MC2y	MC2y	MC2y
16	MC2x	MC2x	MC2x

Tabelle B.5.: Kritische Fehlersequenzen für alle Betriebsmodi

Schritt	N1	N5	N7
1	GPS LNA1+Antenna1	GPS LNA1+Antenna1	MFS Fluxgate2
2	ASC DPU2 High Res. / **GPS Receiver1**	MC1y	GPS Receiver1
3	GPS LNA2+Antenna2	GPS LNA2+Antenna2	GPS LNA2+Antenna2
4	MFS Fluxgate1 / **MC1y**	MFS Fluxgate1	MFS Fluxgate1
5	CSS RearHead2	CSS RearHead2	CSS RearHead2
6	ASC DPU1 High Res.	ASC DPU1 High Res.	ASC DPU1 High Res.
7	ASC DPU2 Low Res.	ASC DPU2 Low Res.	ASC DPU2 Low Res.
8	CSS Chipset1 High Res.	CSS Chipset1 High Res.	CSS Chipset1 High Res.
9	CSS RearHead1	CSS RearHead1	CSS RearHead1
10	CSS FrontHead2	CSS FrontHead2	CSS FrontHead2
11	CSS Chipset2 Low Res.	CSS Chipset2 Low Res.	CSS Chipset2 Low Res.
12	ASC DPU1 Low Res.	ASC DPU1 Low Res.	ASC DPU1 Low Res.
13	MC1x	MC1x	MC1x
14	RW2	RW2	RW2
15	MC2y	MC2y	MC2y
16	MC2x	MC2x	MC2x

Tabelle B.6.: Detaillierte Kennzahlen in Varianten des Rekonfigurationsraums für alle Betriebsmodi

	Vollständiger Graph								Minimierter Graph							
Rekonfig.	Quelle	Ziel	Erreichbar	Ungeführt	Priorisiert	Pfadlänge	Kosten	Qualität	Quelle	Ziel	Erreichbar	Ungeführt	Priorisiert	Pfadlänge	Kosten	Qualität
N1.1	4701	4519	209	102	62	3	22	0,0366	4701	4470	22	3	3	3	8	0,0075
N1.2	4519	4745	209	18	12	12	6	-0,0056	4470	4745	22	1	1	9	8	0,0236
N1.3	4745	4316	209	9	6	13	4	-0,0335	4745	4316	17	5	4	13	4	-0,0335
N1.4	4316	4406	209	6	3	14	2	-0,0039	4316	4406	17	3	2	14	2	-0,0039
	4701	**4406**	**836**	**135**	**83**	**16**	**34**	**-0,0064**	**4701**	**4406**	**78**	**12**	**10**	**16**	**22**	**-0,0064**
N7.1	4567	4705	209	102	38	3	26	-0,0020	4567	4338	20	3	3	3	8	-0,0505
N7.2	4705	4338	209	30	9	9	18	-0,0485								
N7.3	4338	4401	209	18	4	12	8	-0,0008	4338	4401	20	6	2	12	8	-0,0008
N7.4	4401	4381	209	9	2	13	4	-0,0226	4401	4381	20	6	2	13	4	-0,0226
N7.5	4381	4362	209	6	1	14	2	-0,0057	4381	4362	20	4	1	14	2	-0,0057
	4567	**4362**	**1045**	**165**	**54**	**16**	**58**	**-0,0797**	**4567**	**4362**	**80**	**19**	**8**	**16**	**22**	**-0,0797**
N15.1	4726	4745	209	30	14	9	18	0,0233	4726	4450	22	1	1	9	8	-0,0104
N15.2									4450	4401	20	6	4	12	6	0,0070
N15.3	4745	4316	209	9	5	13	4	-0,0344	4401	4316	20	6	4	13	8	-0,0077
N15.4	4316	4406	209	6	3	14	2	-0,0132	4316	4406	17	3	2	14	2	-0,0132
	4726	**4406**	**627**	**45**	**22**	**16**	**24**	**-0,0244**	**4726**	**4406**	**79**	**16**	**11**	**16**	**24**	**-0,0244**
					836	**115**	**53**	**16**	**38,7**	**-0,0368**	**79**	**15,7**	**9,7**	**16,0**	**22,7**	**-0,0368**

B.2. Aufbereitete Datensätze

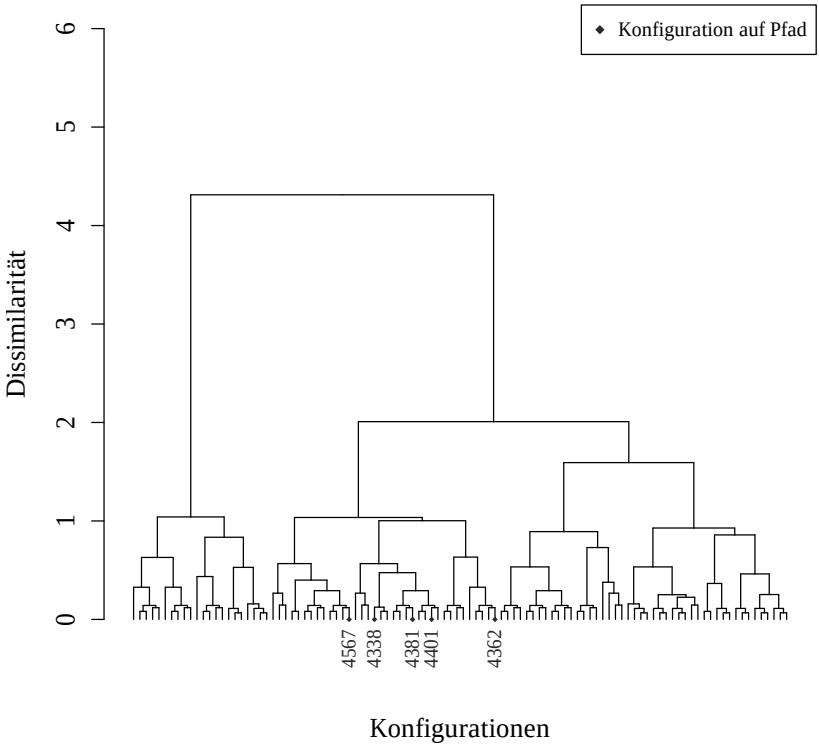


Abbildung B.1.: Dissimilarität der Qualitätsattribute des Betriebsmodus N7

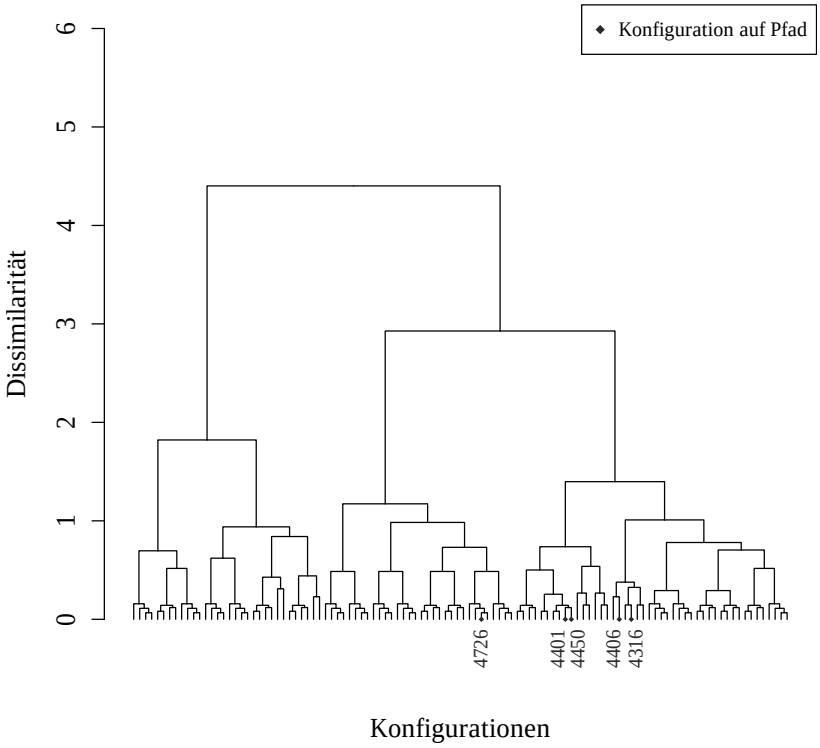


Abbildung B.2.: Dissimilarität der Qualitätsattribute des Betriebsmodus N15

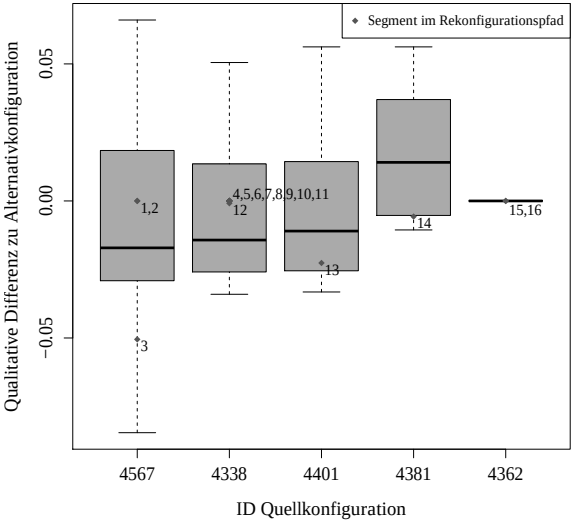


Abbildung B.3.: Verteilung der Qualitätsdifferenzen des Betriebsmodus N7 entlang des Rekonfigurationspfads

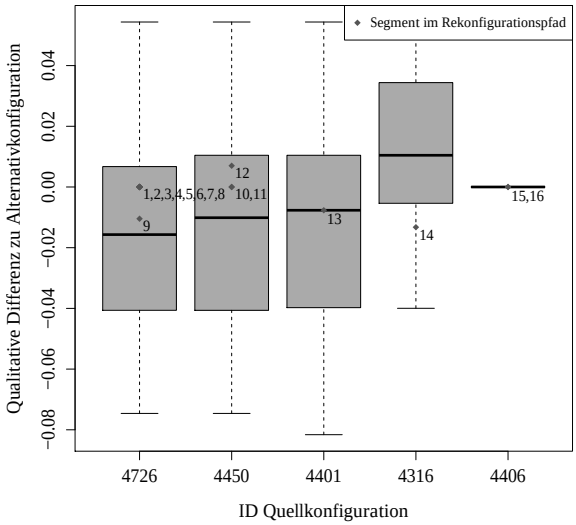


Abbildung B.4.: Verteilung der Qualitätsdifferenzen des Betriebsmodus N15 entlang des Rekonfigurationspfads

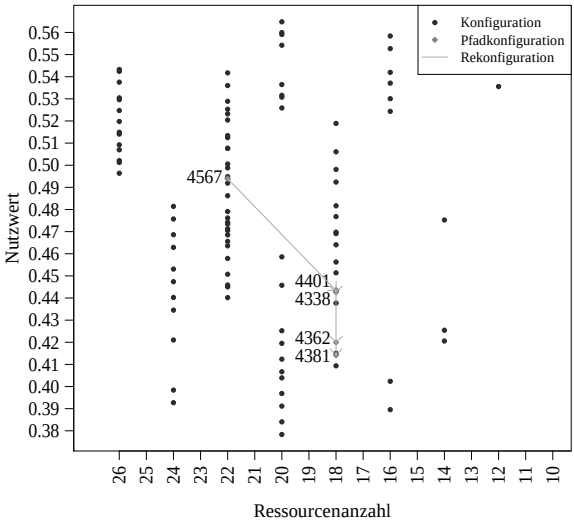


Abbildung B.5.: Streuung der Gesamtqualitäten und Ressourcenmengen des Betriebsmodus N7

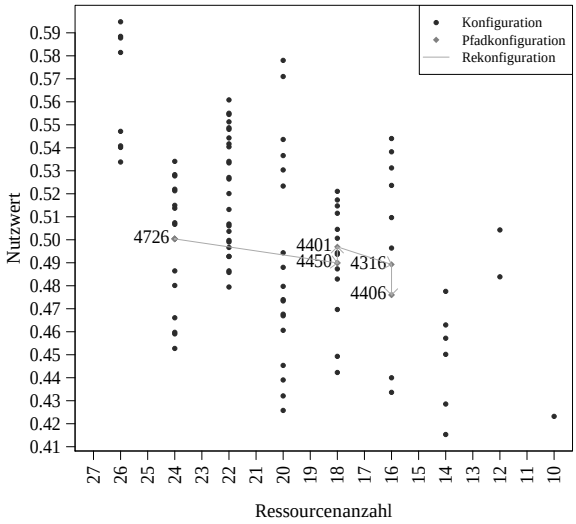


Abbildung B.6.: Streuung der Gesamtqualitäten und Ressourcenmengen des Betriebsmodus N15

C. Leitfaden der Interviews mit den Domänenexperten

C.1. Einleitung

Notiz: Folien zu allgemeinen Informationen werden als Begleitmaterial genutzt.

Vielen Dank, dass Sie sich für ein Gespräch zu wissenschaftlichen Zwecken circa 60 bis 90 Minuten Zeit nehmen. Die Erkenntnisse aus Ihren Aussagen möchte ich zur Schärfung des Kontextes und Bemessung der industriellen Akzeptanz meines Promotionsthemas verwenden. Der rechtlichen Vollständigkeit wegen möchte ich Ihnen vorab das Format und die Rahmenbedingungen dieses Gesprächs kurz erläutern.

Zur groben Strukturierung habe ich einen inhaltlich aufbauenden Interview-Leitfaden vorbereitet. Das Interview ist in fünf Schwerpunkte unterteilt, zusätzlich einer Einleitungs- und Abschlussphase. Bei weiterführendem Gesprächsbedarf sind jedoch jederzeit Abweichungen möglich. Insbesondere bitte ich Sie bei missverständlichen Fragestellungen um Erklärung zu bitten. Sollten die von mir vorbereiteten Fragen aus firmenpolitischen, rechtlichen oder persönlichen Gründen teilweise oder gar nicht beantwortet werden können, bitte ich ebenfalls um entsprechenden Hinweis. Die Auswertung wird entsprechende Restriktionen stillschweigend berücksichtigen.

Zur ausführlichen Auswertung bitte ich Sie um Ihre Erlaubnis zur Aufzeichnung des Gesprächs. Die Auswertung geschieht ausschließlich im Rahmen meiner Dissertation und unterliegt dem Datenschutz. Dabei wäre es sachdienlich, wenn ich ausgewählte Aussagen anonymisiert als Expertenmeinung zitieren dürfte. Wie stehen Sie dazu?

- Ja: Vielen Dank dafür. Ich bitte Sie die Veröffentlichungsgenehmigung durchzulesen und gegenzuzeichnen.
- Ja, aber…: Gerne lasse ich Ihnen die entsprechenden Textpassagen meiner Dissertation vorab zukommen (nachträgliche Autorisierung).
- Nein: Würde eine vorherige Zustellung der entsprechenden Textpassagen meiner Dissertation, mit Möglichkeit einer Zensur, Ihre Meinung dazu ändern? (nachträgliche Autorisierung mit Auflagen)

Info: Veröffentlichungsgenehmigung wird vorgelegt, nach Unterzeichnung wird die Tonbandaufnahme aktiviert.

C.2. Thematische Vorstellung

Info: Folien zu dem thematisierten Ansatz werden als Begleitmaterial genutzt.

Zunächst möchte ich Ihnen den Ansatz vorstellen welchen ich in meiner Promotion verfolge. Weiterhin möchte ich meine Annahmen sowie die Rolle der Fallstudie TET-1 ACS für meine Arbeit erläutern.

C.3. Inhaltlicher Teil des Interviews

C.3.1. Persönliche Situation

Lassen Sie uns zunächst über Ihre Interessen und Erfahrungen sprechen, damit ich unser Gespräch darauf ausrichten kann.

1. Bitte erläutern Sie kurz Ihren theoretischen und praktischen Hintergrund.
2. Bitte stellen Sie kurz dar welche Erfahrungen Sie in der Domäne der Entwicklung von Raumflugkörpern bisher gesammelt haben.
    a) An welchen Projekten waren Sie beteiligt?
    b) Stehen Sie generell mit Universitäten bzgl. Forschung in Kontakt?

3. Welche Rolle haben Sie bei Entwicklung und Betrieb des Satellitenbus TET-1 und dessen Nachfolger BIROS (oder TET-X bzw. TET-XL)?
    a) Liegt Ihr Fokus eher auf Software- oder Hardwareentwicklung?
    b) Waren Sie vorrangig an der Systemintegration oder an der EEE-Bauelemente-Entwicklung beteiligt?
    c) Kennen Sie sich mit den Telemetrie-Daten Ihrer Smart Components aus?
    d) Hatten Sie Einblicke in die Missionsplanung?
4. Offiziell ist Astrofactum für den Betrieb verantwortlich, haben Sie dennoch Kenntnisse über die Wartungsprozesse innerhalb des OOV-Programms?
    a) Kennen Sie die Prozesse für Software-Updates?
    b) Hatten Sie Einsicht in Wartungsprotokolle?
    c) Waren Sie unmittelbar an der Umsetzung von Änderungsanfragen (Change Requests) beteiligt?

C.3.2. Konfigurierbarkeit von Systemen

In Systemen mit hoher Fehlertoleranz durch multiple Redundanzen, wie dem TET-1 Satellitenbus, liegt inhärent ein großer Entwurfsraum mit einem hohen Grad an **Konfigurierbarkeit** vor.

In dieser Untersuchung umfasst eine **Systemkonfiguration** die zum aktuellen Zeitpunkt genutzten Softwaremodule sowie deren zur Ausführung notwendigen Hardwareressourcen. Ressourcen können dabei berechnende Einheiten (ECUs, …) sowie Datenquellen/–senken (vor allem Sensoren/Aktuatoren) sein. Die **Validität einer Konfiguration** ergibt sich aus den Ein- und Ausschlussbeziehungen der Hardwareressourcen (das heißt deren Kompatibilität zueinander) sowie der funktionalen und qualitativen Erfüllbarkeit geltender Missionsanforderungen. Der Begriff der Konfigurierbarkeit richtet sich hier somit vor allem nach den verfügbaren Sensoren und Aktuatoren und insbesondere nicht nach der Vielfalt an Modi der Lageregelung.

1. Die genannten Begriffe sind domänenspezifisch mehrfach belegt. Können Sie meiner Definition folgen oder sollten wir andere Begrifflichkeiten verwenden?
    a) Systemkonfiguration?
        i. Softwarekonfiguration?
        ii. Hardwarekonfiguration?
    b) Konfigurierbarkeit? Parametrisierbarkeit?
    c) Hardwareressourcen hier vorrangig redundante Sensoren und Aktuatoren
2. Sind Sie mit dem Missionsmanöver „Kepler's Second Light (kurz: K2)" des Weltraumteleskops Kepler (NASA) zur Auffindung von Exoplaneten vertraut?
    a) Ja: Gut, dann möchte ich darauf etwas näher eingehen.
    b) Nein:
        i. Durch den Ausfall von zwei Reaktionsrädern im Mai 2013 konnte Kepler nicht mit mehr präzise ausgerichtet werden.
        ii. Die Hauptmission wurde Mitte August 2013 beendet
        iii. Eine außergewöhnliche Modifikation der Systemkonfiguration erlaubte Fortsetzung, allerdings mit qualitativ abgestufter Mission ab Mai 2014
        iv. Umgang der wegfallenden Reaktionsräder an Bild erläutern.
3. Können Sie sich vorstellen, dass solch eine Systemkonfiguration bereits im Entwicklungsprozess ermittelt worden wurde?
    a) Ja:
        i. Woran machen Sie das fest?
        ii. Sollten Strategien solcher Ausprägung zur Wiederverwendung in zukünftigen Missionen aufgegriffen werden?

b) Nein: Aus welchen Gründen erscheint Ihnen dies unrealistisch?

C.3.3. Exploration des Rekonfigurationsraums

Kommen wir zur Festlegung von **erwarteten** Systemkonfigurationen in einem **üblichen** Entwicklungsprozess. Lassen Sie uns dabei bitte zunächst über das Vorgehen zur Spezifikation von Konfigurationen sprechen.

1. In welcher Form werden Konfigurationen im Entwicklungsprozess festgelegt?
    a) Spezifisch: In Form von Anforderungen ausgehend von den Missionszielen?
    b) Abgeleitet: Aus den möglichen Kombinationen redundanter Baugruppen?
    c) Gar nicht: Es gibt keine Konkretisierung von Konfigurationen.
2. Nach welcher Maßgabe werden Redundanzen festgelegt?
    a) Fehlerausfallraten und Fehlerbaumanalysen?
    b) Missionskritische Bauteile?
    c) Freie Entwurfsentscheidung aus Historie?

Gehen wir im nächsten Abschnitt von einem konkreten Ausfall in einer Redundanzgruppe im TET-1 aus.

1. Wie würde das FDIR des ACS im Idealzustand des Systems mit einem Ausfall während eines Experiments umgehen?
2. Sind die Redundanzen im TET-1 immer in Gruppen symmetrisch, d.h. identisch-austauschend, definiert?
    a) Symmetrisch/identisch: Gilt eine alternative Ressource also auch dann als redundant, wenn sie nur einen Teil der Funktionalität des Originals erfüllt?
    b) Auch asymmetrisch/teilnachbildet: Welche Rolle spielen hier zum Beispiel Ansätze der Sensorfusion?

Lassen Sie uns zur weiteren Vertiefung ein simplifiziertes Beispiel einer asymmetrischen Redundanzbeziehung aus dem Automobilbereich aufgreifen. Ein Notrad ersetzt bei einer Reifenpanne zwar kurzfristig ein defektes Rad, dies hat jedoch deutliche Auswirkungen auf die Fahrdynamik des Fahrzeugs. Neben abweichenden funktionalen Eigenschaften (Dimensionen, Gummimischung) weichen auch qualitative Attribute (Fahrtgeschwindigkeit, Reichweite) ab. Dies bedingt eine Änderung des Betriebsmodus des Fahrzeugs. Konkret findet eine qualitative Degradation auf max. 80 km/h Fahrtgeschwindigkeit sowie eine Einschränkung der Fahrstreckenflexibilität statt.

1. Sind in Abhängigkeit der Redundanzsituation solche qualitative Abstufungen im System vorgesehen? (Vertiefung Sensorfusion-Analogie)
    a) Ja: Ist es aktuell möglich die ACS-Modes darüber zu parametrisieren?
    b) Nein: Gilt ein System, zum Beispiel das TET-1 ACS, also als grundsätzlich entweder funktional oder nicht-funktional beziehungsweise „(un)healthy“?
2. Werden für Missionen konkrete qualitative Anforderungen aufgestellt?
    a) Ja: Erlaubt dies allein eine frühzeitige Prognose der Erfüllbarkeit von Missionszielen in Abhängigkeit der Konfiguration?
    b) Nein: Ist der Prozess rein funktionsorientiert?

C.3.4. Ablauf des Wartungsprozesses

Nachdem wir diskutiert haben, wie die Konfigurierbarkeit des Systems mit Redundanzen zusammenhängt, würde ich gerne mit Ihnen über Wartungsprozesse sprechen. Dabei steige ich wieder mit einem Bericht zum TET-1 aus dem eoPortal ein.

Zu Beginn der Betriebsphase des TET-1 traten unerwartete Temperaturanstiege in den Nickel-Hydrogen-Batterien auf.

1. Sind Ihnen die Umstände geläufig?
    a) Ja: OK
    b) Nein: Dann lassen Sie mich kurz reflektieren: Der batteriegestützte Flugbetrieb im Erdschatten war sicherzustellen. Zur Behebung des Temperaturproblems wurden einige Ladestrategien erprobt. Zuletzt wurde die Ansteuerung der Solarpanel-Arrays durch ein Software-Update angepasst.
2. Wie würde der Wartungsprozess idealtypisch von der Fehleridentifikation bis zu dessen Behebung ablaufen?
    a) Welche Rolle spielen dabei „on-ground“- und „in-orbit“-Tests?
    b) Gibt es in der Entwicklungsphase entsprechende Standardprozeduren, deren Messdaten im Fehlerfall herangezogen werden?
3. Hatte das FDIR des TET-1 die Batterieprobleme selbstständig identifiziert?
    a) Waren die notwendigen Informationen in der vorhandenen Housekeeping-Daten enthalten?
        i. Ja: Ok
        ii. Nein: Mussten zusätzliche Diagnosen durchgeführt werden?
4. Wie wurde die Lösungsstrategie für das Problem ermittelt?
    a) Alternativen:
        i. Handelte es sich um eine strukturierte Ursache-Wirkungsanalyse auf Grundlage der Hardwarearchitektur des Satelliten?
        ii. Wurden Referenzsysteme hinsichtlich des Fehlerverhaltes untersucht? (Hinweis auf Kepler Space Teleskop im Bericht geben)
        iii. Wurde die Lösung ausschließlich durch Expertenwissen ermittelt?

5. Wie werden solche Änderungen der Systemkonfiguration aktuell protokolliert?
    a) Speziell strukturiert als Working Dokument?
    b) Housekeeping-Datensammlung?
6. Werden routinemäßige Selbstdiagnosen über Telekommandierung angestoßen?
    a) Ja:
        i. Insbesondere nach dem Einspielen eines Software-Updates?
        ii. Bedarf die Diagnose eine ständige Verbindung zum System?
    b) Nein: Wie werden Software-Updates auf Funktionalität überprüft?

Kommen wir im nächsten Abschnitt zu den natürlichen Beschränkungen die in der Fernwartung von Software-/Hardwaresystemen gelten. Dabei möchte ich gerne über generelle Richtwerte sprechen und nur am Rande über das konkrete Batterieszenario.

1. Welche Prozesscharakteristika beschränken die Fernwartung aus Ihrer Sicht vorrangig?
    a) Personalaufwand?
    b) Kommunikation?
    c) Systemzugriff?
    d) Sonstiges?
2. Beschränkung durch Höhe des Personalaufwands:
    a) Wie viele Personen sind ca. an einem Software-Update beteiligt?
    b) Wie viel Zeit nahmen vergangene Software-Updates durchschnittlich ein?

3. Beschränkungen durch unterbrochene Kommunikationsverbindung: (Bild zu den Bodenstationen mit Kontaktanzahl zeigen)
    a) War die Häufigkeit der Kontakte pro Passage immer ausreichend?
        i. Ja: Also die üblichen 4x/Tag ? (1x Weilheim für TC's, 3x Neustrelitz für Payload data dump, Telemetrie)
        ii. Nein:
            A. Wurde die Aufteilung geändert?
            B. Wurden hohe Kosten für zusätzliche kommerzielle Bodenstationen in Kauf genommen?
    b) Die durchschnittliche Dauer eines Kontakts zum TET-1 beträgt 7 bis 9 Minuten pro Passage. Beschränkt dabei der Uplink zum TET-1 das Durchführen eines Software-Updates?
        i. Ja: Welche Maßnahmen werden dann ergriffen?
        ii. Nein:
            A. Welchen Umfang hat die Firmware des ACS in etwa in KB?
            B. Genügt die Übertragungsrate von 4 kbit/s für einen Kontakt?
4. Beschränkung des Systemzugriff durch Telemetrie:
    a) Ihre Smart Components (z.B. das RW90) liefern sehr umfangreiche Telemetriedaten. Wie detailliert sind die Housekeeping-Daten?
        i. Fein: Bitte geben Sie ein Beispiel.
        ii. Grob: Warum werden nicht mehr Daten geliefert?
    b) Ist eine Art betriebsbegleitender Debug-Modus im Betrieb vorgesehen?
        i. Ja: Wie läuft dieser ab?

ii. Nein: Können Telemetriedaten in unterschiedlichen „Log-Levels“abgefragt werden?

Als Nächstes möchte ich über die Bedeutung von Modellen im Wartungsprozess sprechen. Die Controller-Software der Reaktionsräder des TET-1 ACS wurde modellbasiert entworfen.

1. Werden diese und weitere Modelle ebenfalls für Vorrausagen im Wartungsprozess aktuell eingesetzt?
   a) Fehlerbäume: Nur zur Definition der FDIR-Implementierung?
   b) Hardwaremodelle: Also nicht nur zur Kalibrierung?
   c) Softwaremodelle: Wird der verbleibende Raum alternativer Konfigurationen nach einem Software-Update erhoben?
      i. Ja: Wie wird dies gemacht?
      ii. Nein: Kann dies nicht zu neuen Problemen führen?
   d) Weitere?
2. Sind die bisherigen Änderungen am System auf Architekturebene sichtbar?
   a) Ja: Bis zu welchem Grad?
   b) Nein: Sind Strukturanpassungen also in Verzweigungen im Code verborgen?

Sie sind neben der Entwicklung u.a. auch für den **Ground Support Test** verantwortlich. Bitte lassen Sie uns in diesem Zusammenhang über die Verwendung von SiL- und HiL-Ansätzen sprechen.

1. Werden Änderungen vor deren Anwendung auf der Erde modellartig erprobt?
   a) Ist dies nur in der Entwicklungsphase am ACS-Teststand (HiL) möglich…oder ebenfalls simulativ mit SiL und Hardwaremodellen im Betrieb?
2. Werden weiterhin Modelle genutzt die den Systemkontext nachbilden?
   a) Schwerkraft, Strahlung, Magnetismus?

C.3.5. Autonomie und Transparenz im Wartungsprozess

Kommen wir in der nächsten Passage zu Sicherheitsmechanismen, die den reibungslosen Betrieb gewährleisten. So kam es im Laufe der OOV-Mission zu einigen Anpassungen die u.a. die Autonomie des Systems erhöhten. Der Safe-Mode schützt das System im Fehlerfall vor dem Komplettausfall von Kommunikation und Energieversorgung.

1. Der TET-1 wird als sehr zuverlässiges System beworben. Bleibt ein Ausfall dennoch weiterhin kritisch?
    a) Welche Bedeutung hatte die lückenlose Durchführung der 11 Experimente?
    b) Kam es zu Unterbrechungen im produktiven Betrieb?
2. Wird der Safe-Mode ebenfalls während eines Software"=Updates aktiviert?
3. Sollte der Safe-Mode langfristig als Konfigurationsalternative dienen oder nur kurzfristig?
    a) Gibt es Seiteneffekte im Safe-Mode die langfristig das System oder dessen Erreichbarkeit schädigen? (Abnutzung Akkus oder Aktuatoren?)
4. Wurde eine Anpassung des Safe-Mode an sich vorgenommen?

Über die Erfordernisse und Chancen autonomer Erweiterungen möchte ich nachfolgend sprechen. Stellen Sie sich vor, der FDIR-Mechanismus des TET-1 initiiert selbstständig nach einem Sensor-/Aktuatorausfall eine Rekonfiguration. Gehen wir dabei von zwei Anpassungsszenarien aus. Bitte beurteilen Sie jeweils die Akzeptanz des Wartungspersonals für die autonome Anpassung der Konfiguration:

1. a. Beurteilung bei gleichbleibender Systemqualität; unveränderte Fortführung der Mission möglich.
2. b. Beurteilung bei eklatant sinkender Systemqualität, eingeschränkte Fortführung der Mission möglich.

3. Wäre alternativ eine Entscheidungsunterstützung zur Beschränkung des Alternativenraums ohne automatische Anpassung denkbar (passives Expertensystem)?
4. Was sollte eine Beschreibung des Rekonfigurationsprozesses enthalten?
    a) Welche Informationen würden die Nachverfolgung des Vorgangs effektiv unterstützen? (Kernel-Meldungen, syslogs, ...)
5. Welche Auswirkungen interessieren Sie bei Änderung der Konfiguration besonders?
    a) Umschalten welcher Hardware?
    b) Qualitative Änderungen?
    c) Datenstandänderung?
6. Das ACS ist in der Lage autonom in den Safe Mode zu wechseln, wenn ein Modus eine bestimmte Zeit nicht angefahren werden kann. Auf welchen Beobachtungen wird diese Rekonfigurationsentscheidung getroffen?
    a) Surveillance-Monitor einzelner Ressourcen (Watch Dog Timer)?
    b) Sonstiges?
7. Werden andere Modi in Abhängigkeit eines Kontextes autonom angefahren?
8. Gab es weitreichende Änderungen der Systemautonomie im Betrieb?
    a) Ja: Erleichtern diese zugleich die Wartungsarbeiten?
    b) Nein: Gibt es entsprechende Konzepte für BIROS (TET-X oder TET-XL)?

C.3.6. Anwendbarkeit des Ansatzes

Lassen Sie uns jetzt wieder auf meinem Ansatz zurückkommen. Meine Arbeit stellt die Systemarchitektur und qualitative Analysen in den Vordergrund. Ich folge dabei einem Entwurfsparadigma, welches die Simulierbarkeit von Architekturen ermöglicht. Dies bedarf eine umfassende Architekturmodellierung, rudimentäre Verhaltensspezifikationen sowie eine frühe Prognose von Systemqualitäten.

Info: Abbildung zum Palladio-Konzept als Begleitmaterial.

1. Bitte beurteilen Sie die Nähe der produktiven Implementierung des ACS zu dessen Ausgangsarchitektur im Entwurf.
2. Können Sie sich vorstellen, dass Architekturen eine zentrale Rolle im Entwicklungsprozess einnehmen könnten?
    a) Ja:
        i. Werden Architekturen zur Analyse von Interaktionsbeziehungen und Verteilungen von Software-Modulen genutzt?
        ii. Werden Wirkungsketten zwischen Sensoren und Aktuatoren architekturbasiert optimiert?
        iii. Deckt sich die Realisierung im Code mit den Konzepten des Architekturentwurfs?
    b) Nein:
        i. Wird die Softwarearchitektur rein deskriptiv zur Dokumentation eingesetzt oder auch zur Analyse?

Nehmen wir als Beispiel für zwei Konfigurationen die Steuerungsvarianten des Reaktionsrad-Controllers. Im ersten Fall werden drei Reaktionsräder mit höherer Geschwindigkeit angesteuert; im zweiten Fall vier Räder bei verringerter Geschwindigkeit.

1. Sind beide Varianten des Controllers auf Architekturebene sichtbar oder nur im Code sichtbar?
    a) Sichtbar: OK

   b) Nicht sichtbar:

      i. Besteht ein Interesse daran Phänomen der Rekonfiguration auf Architekturebene sichtbar zu machen?

         A. Ja: Was wäre Ihnen besonders wichtig?

         B. Nein: Wo sehen Sie Hürden?

2. Können Sie sich vorstellen Informationen aus dem Architekturentwurf im Betrieb zur Analyse in der Wartung zu nutzen?

3. Sollte dabei eine Nachverfolgbarkeit in Konfigurationsschritten auf Architekturebene unterstützt werden?

   a) Ja/weiß nicht: OK

   b) Nein: Weshalb nicht?

Gehen wir vom Architekturgedanken wieder weg und näher auf qualitative Degradationen von Entwurfsalternativen ein.

1. Werden konkrete Konfigurationsalternativen bereits im Entwurf gegeneinander abgewogen?

   a) Orientiert sich die Entwicklung von Bauelementen eher an funktionalen und qualitativen Zielwerten?

   b) Wird das System frühzeitig gegen Missionsanforderungen gemessen (vgl. Special Purpose Satellite System)?

   c) Findet eine qualitative Bewertung und Auswahl statt?

2. Welchen Anteil hat die Definition von Fehlererkennung und-behandlung momentan im Entwurf? (FTA und FDIR-Definition)

3. Ist ein Mehraufwand zur frühzeitigen Konfigurationsraumanalyse in Entwurf realistisch umsetzbar?

   a) Ja: OK

   b) Nein: Und wenn damit die FDIR-Aufwände teilweise substituiert werden?

4. Sind die notwendigen Daten verfügbar bzw. mit Datenblättern vorausberechenbar (Annahme zur Weltabgeschlossenheit)?
    a) Ja: OK
    b) Nein:
        i. Ist der Rauschfaktor experimentell messbar oder prognostizierbar?
        ii. Welche Rolle spielen Erfahrungswerte aus vorherigen Missionen?
5. Ist die Unschärfe beim Einsatz von Meta-Heuristiken zur Identifikation gültiger Konfigurationen tragbar für Analysen in Ihrer Domäne?
    a) Ja: OK, also wäre eine strukturiert strichprobenartige Untersuchung ausreichend?
    b) Nein: Wären alternativ exakte Vergleiche notwendig? Bitte bedenken Sie dabei den großen Zustandsraum und die erschöpfenden Suchaufwände.
6. Wäre eine verstärkt architekturbasierte Dokumentation in die Prozesse der Entwicklung und Wartung integrierbar?
7. Bitte wägen Sie allgemein die Vor- und Nachteile meines Ansatzes zur Unterstützung des Wartungsprozesses ab.

C.3.7. Schluss

Wir kommen nun zum Abschluss unseres Gesprächs. Lassen Sie mich kurz zusammenfassen. Wir haben gesprochen über Konfigurierbarkeit, Wartungsprozesse, Autonomie und Transparenz in der Wartung und zuletzt über die Akzeptanz meines Ansatzes.

Wenn Sie noch Nachfragen, Ergänzungen oder Korrekturen zu einzelnen Punkten haben, können wir diese gerne in einem späteren Telefonat oder per Email besprechen.

Ich möchte mich noch einmal ganz herzlich für Ihre Beteiligung und Diskussionsbereitschaft bedanken. Ich bin sehr zuversichtlich, dass ich interessante Erkenntnisse aus Ihren Aussagen ziehen kann. Gerne lasse ich Ihnen meine Forschungsergebnisse nach Fertigstellung meiner Dissertation zukommen.

D. Transkript der Interviews mit den Domänenexperten

D.1. Tabellarische Zusammenfassungen

Tabelle D.1.: Antworten zu Frage 1 aus Konfigurierbarkeit von Systemen

Kat. Konfigurierbarkeit von Systemen		Frage 1
Analysedim. Modellannahmen		Analyse irrelevant
Schwerpunkte	Begriffsdefinitionen von Softwarekonfiguration, Hardwarekonfiguration, Konfigurierbarkeit und Parametrisierbarkeit	
1. Antw. 0:51:18	Begrifflichkeiten akzeptiert und Festlegung als dringend notwendig beschrieben.	
2. Antw. 0:14:00	Begrifflichkeiten akzeptiert und insbesondere Systemkonfiguration und Architektur voreinander abgegrenzt.	

Tabelle D.2.: Antworten zu Frage 2 aus Konfigurierbarkeit von Systemen

Kat. Konfigurierbarkeit von Systemen		Frage 2
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Kenntnis von Kepler's Second Light (K2)	
1. Antw. 0:57:00	Nein, Kurzvorstellung erfolgt.	
2. Antw. 0:17:55	Bekannt, aber nochmal reflektiert.	

Tabelle D.3.: Antworten zu Frage 3 aus Konfigurierbarkeit von Systemen

Kat. Konfigurierbarkeit von Systemen		Frage 3
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Einschätzung der frühzeitigen Identifikation einer zu K2 ähnlichen Lösung	
1. Antw. 0:57:00	Wissen aus Vorgängermissionen wird genutzt. Beispielsweise sind beim dem Vorgänger vom TET-1 (BIRD) die Reaktionsräder nach und nach ausgefallen. Es kam dadurch zu Agilitäts- und Genauigkeitsverlusten, aber man konnte prinzipiell weiterarbeiten. So wurde anschließend über zehn Jahre lang ein Betrieb ausschließlich mit Magnetspulen realisiert. Im TET-1 wurde so natürlich eine Stabilisierung zur Sonne hin mit Magnetspulen als Rückfallebene per Software-Update implementiert. Dies ist ein Sonderfall, aber eine Übertragung auf die normale Lageregelung wäre möglich. Für optische Nutzlasten reicht dies aber in der Regel nicht aus. Die Berücksichtigung und Nutzung des Sonnendrucks ist üblich. Bei Satelliten auf höheren Umlaufbahnen (als TET-1) werden teilweise die Sonnensegel phasenweise gekippt um Treibstoff einzusparen. Eine Kombination mit zwei Reaktionsrädern (wie bei K-2) ist aber nicht üblich.	
2. Antw. 0:21:02	Die Exploration nach möglichen Alternativen kommt auf die Missionsvorgabe an und mit wie vielen Fehlern zu rechnen ist. Grundsätzlich ist eine 1-Fehlertoleranz Standard, weitere Ausfälle werden in der Regel nicht betrachtet. Das Vorhandensein einer weiteren Lösung ist ein gutes Argument, aber nicht erforderlich um übliche Anforderungen zu erfüllen. Beim TET-1 wäre ein vergleichbares Manöver nicht möglich, da der Sonnendruck im niedrigen Erdorbit zu gering ist. Die meisten Missionen wären auch ohne Reaktionsradsystem möglich, allerdings wäre die Agilität für den Normalbetrieb zu gering. Mit einem langen Zeitraum zur Ausrichtung auf einen bestimmten Punkt auf der Erde, wäre ein Notbetrieb noch möglich. Dabei wäre formal die Missionsanforderung (hohe Agilität) nicht mehr erfüllt.	

Tabelle D.4.: Antworten zu Frage 1 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 1
Analysedim. Aufbau Entwurfsraum		Analyse in 6.5.8.8, **FF2.4**
Schwerpunkte	Festlegung von Konfigurationen	
1. Antw. 1:06:00	Die Auswahl und das Design der Hardware richtet sich nach dem Missionsziel. Anforderungen (zum Beispiel optische Messungen) geben die Sensoren und Aktuatoren vor. Die Anzahl richtet sich weiterhin nach den Zuverlässigkeitsanforderungen. Bei TET-1 wurden die Anforderungen aus der Summe der 11 Experimente gebildet. Mögliche weitere Szenarien wurden nicht untersucht. Dies wird erst geprüft, wenn ein Problem auftritt.	
2. Antw. 0:27:05	Konfigurationen werden spezifisch festgelegt und richten sich nach dem Missionsziel. An den Anforderungen an das Redundanzkonzept leiten sich die Konfigurationen ab. Mögliche Konfigurationen sind aus dem Vorwissen bekannt, es kommt zu keinen neuen Kombinationen.	

Tabelle D.5.: Antworten zu Frage 2 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 2
Analysedim. Aufbau Entwurfsraum		Analyse in 6.5.8.3, **FF1.3**
Schwerpunkte	Festlegung von Redundanzen	
1. Antw. 1:09:31	Die Anforderung für TET-1 war eine 1-Fehlertoleranz, dies wird in Fehlerbaumanalysen bestätigt. Die Beurteilung von heißen und kalten Redundanzen richten sich nach der Masse des Systems. Die Tetraeder-Anordnung des Reaktionsradsystems ist ein Kompromiss zwischen Fehlertoleranz, Masse und Energie. Die Reaktionsräder bieten mehr Funktionalität als gefordert, erlauben daher aber auch einen 3-Rad-Betrieb. Die Historie (Flugerfahrung) beeinflusst die Auswahl.	
2. Antw. 0:29:05	Redundanzen legt der Auftragsgeber fest. Beim TET-1 war jedes Element als redundant gefordert, unabhängig davon wie kritisch es ist. Bei dem Bord-Computer wurde sogar eine vierfache Redundanz vorgesehen. Ausfallraten müssen bestimmt werden. Die Methode ist umstritten, aber gilt als Standard in der Raumfahrt. Manche Kennwerte [aus Bauteilspezifikationen] müssen in Missionen auch nachgewiesen werden.	

Tabelle D.6.: Antworten zu Frage 3 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 3
Analysedim. Prozessverständnis		Analyse in 6.5.8.2, **FF1.2**
Schwerpunkte	Ablauf der Fehlerbehandlung im Idealzustand des Systems	
1. Antw. 1:12:50	Es werden Erwartungswerte aus Modellen mit realen Messwerten verglichen. Tritt eine Abweichung aus, wir der Sensor "fehlermarkiertünd der Modellwert wird zunächst ausschließlich benutzt. Beispiel Sternenkamera: Wenn die Sternenkamera von der Sonne geblendet wird, berechnet das System eine Zeit lang die Lage modellbasiert weiter. Dadurch entsteht ein Drift. Nach Ablauf der Wartezeit wechselt das System in einen anderen Modus (hier: einfache Sonnenausrichtung) um das Problem zu kompensieren. Die Sensoren werden alle parallel betrieben und Ausfälle werden zwischenzeitlich Software-seitig kompensiert. Aktuatorenausfälle werden zunächst ignoriert, solange Redundanzen vorliegen. Sobald diese Bedingung verletzt ist wird der ACS Safe-Modus automatisch angefahren.	
2. Antw. 0:32:00	Frage übersprungen, da ausführlich im vorherigen Interview erläutert.	

Tabelle D.7.: Antworten zu Frage 4 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 4
Analysedim. Aufbau Entwurfsraum		Analyse irrelevant
Schwerpunkte	Homogene und heterogene Redundanzen	
1. Antw.	Funktionsorientiert heterogen, hardwareorientiert ausschließlich homogen.	
2. Antw. 0:32:40	Redundanzen bezeichnen Doppelungen in der Hardware. Theoretische Alternativen sind bekannt (zum Beispiel Berechnung der inertialen Lage mit der Sternenkamera), gelten aber nicht als redundant.	

Tabelle D.8.: Antworten zu Frage 5 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 5
Analysedim. Prozessverständnis		Analyse in 6.5.8.11, **FF4.2**
Schwerpunkte	Existenz qualitativer Degradationen	
1. Antw. 1:19:55	Abstufungen sind vorgesehen, zum Beispiel mit unterschiedlichen Sensoren zur Bestimmung der inertialen Lage. Eine Änderung des Lagereglungsmodus ist an qualitative Bedingungen (Mindestwerte) geknüpft. Anstatt alternativen Sensoren zu nutzen, sind auch modellbasierte Berechnungen möglich. Der Anwender im Bodensegment kann auf Grundlage diverser Flags die erreichbare Qualität der Lageregelung berechnen und dann entscheiden ob diese ausreicht.	
2. Antw. 0:34:43	Qualitative Abstufungen treten mit Bezug auf redundante Komponenten nicht auf.	

Tabelle D.9.: Antworten zu Frage 6 aus Exploration des Konfigurationsraum

Kat. Exploration des Konfigurationsraum		Frage 6
Analysedim. Prozessverständnis		Analyse in 6.5.8.11, **FF4.2**
Schwerpunkte	Rolle qualitativer Anforderungen	
1. Antw. 1:24:00	Missionen geben qualitative Anforderungen vor	
2. Antw. 0:35:30	Konkrete Missionsanforderungen sind als Zahlen aus Dokumenten auslesbar.	

Tabelle D.10.: Antworten zu Frage 1 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 1
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Kenntnis von Batterie-Wartungsvorfall im TET1	
1. Antw. 1:25:00	Die Umstände sind geläufig.	
2. Antw. 0:36:55	Keine Beteiligung am Projekt in der Anfangsphase, daher kein Detailwissen über das Problem.	

Tabelle D.11.: Antworten zu Frage 2 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 2
Analysedim. Prozessverständnis		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Idealtypischer Ablauf des Wartungsprozesses	
1. Antw. 1:27:00	Der Sachverhalt wird festgestellt und die Mission wird fortgesetzt wenn die Einschränkung akzeptabel ist, zum Beispiel nur zeitliche Verzögerung. Beispiel für die beschleunigte Batterieentladung: Es wird nur die Hälfte der Erdobservation durchgeführt und in der Zwischenzeit wird im Bodensegment nach einer Lösung gesucht. Dies vermeidet einen Totalausfall. Entweder wird die Software oder der Missionsbetrieb angepasst. Bei Software-Modifikationen werden vorrangig Algorithmen und Strategien angepasst. Neue Fehlerbehandlungsroutinen und Modi werden eher nicht eingeführt. Änderungen werden möglichst häufig am Boden verifiziert um Seiteneffekte festzustellen.	
2. Antw. 0:38:30	Jegliche Anpassung wird intensiv auf der Erde getestet um zu vermeiden, dass das System nach einem Update versagt. Umwelteinflüsse können dabei nur bedingt nachgebildet werden.	

Tabelle D.12.: Antworten zu Frage 3 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 3
Analysedim. Prozessverständnis		Analyse in 6.5.8.2, **FF1.2**
Schwerpunkte	Ablauf der Diagnose des Fehlers im TET	
1. Antw. 1:29:30	Erfahrungen des Personals bilden die Basis für die Fehleranalyse. Erwartete Fehler werden gezielt geprüft, zum Beispiel durch das Abziehen eines Kabels auf dem Teststand. Die Lokalisierung eines Fehlers wird zum Beispiel durch festgestellt, dass kein Strom mehr fließt und die Baugruppe somit wahrscheinlich defekt ist. Das Batterieproblem wurde durch einen Vergleich von Temperaturen vor und während eines Ladevorgangs festgestellt. Die Zusammenhänge wurden händisch durch Expertenwissen ermittelt.	
2. Antw. 0:39:50	Ein Teststand ist Standard zum Vergleich. Ein Teststand zur Nachbildung der Umwelteinflüsse (hier: TET-1 ACS Teststand) ist selten verfügbar.	

Tabelle D.13.: Antworten zu Frage 4 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 4
Analysedim. Prozessverständnis		Analyse in 6.5.8.4, **FF2.3**
Schwerpunkte	Ermittlung einer Lösungsstrategie	
1. Antw. 1:31:20	Simulationen werden auf der Testplattform durchgeführt.	
2. Antw. 0:41:20	Frage ausgelassen, da in der Phase nicht am Projekt beteiligt.	

Tabelle D.14.: Antworten zu Frage 5 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 5
Analysedim. Prozessverständnis		Analyse in 6.5.8.10, **FF3.4**
Schwerpunkte	Protokollierung der Änderung der Systemkonfiguration	
1. Antw. 1:32:40	Grundsätzlich werden die Daten in Textform festgehalten. Die Fehler sind sehr vielfältig und daher schwer auf eine bestimmte Notation abzubilden. Das Batterieproblem führt zum Beispiel zu einem thermalen Gesamtproblem für den gesamten Satelliten. Im GSOC könnte es Skizzen über die Verfügbarkeit geben, Details sind jedoch nicht bekannt.	
2. Antw. 0:42:00	Je nach Detaillierungsgrad liegen Änderungsinformationen vor, diese sind aber zumeist bezogen auf Bauteilgruppen. Alle Daten zu aggregieren ist schwierig, da abhängig vom jeweiligen Entwickler. Enger Zusammenhalt im Team baut auf regelmäßiger Synchronisation auf. Dies ist in der Domäne möglich, da ein System hochspezialisiert entwickelt wird.	

Tabelle D.15.: Antworten zu Frage 6 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 6
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Existenz routinemäßiger Selbstdiagnosen	
1. Antw. 1:34:45	Nein, Diagnosen werden nur durch Telekommandierung angestoßen. Ein Fehler würde erst bei der Aktivierung einer Komponente auffallen.	
2. Antw. 0:48:50	Diagnosen finden nach einem Update statt und beinhalten einen Checksummen-Test des Uploads und eine Überprüfung ob der Fehler behoben wurde. Dies wird in der Regel zu einem Zeitpunkt mit einem langen Überflug (vielen Kontakten) durchgeführt. Nach der Diagnose wird per Telekommandierung in den normalen Betriebsmodus zurückgewechselt.	

Tabelle D.16.: Antworten zu Frage 7 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 7
Analysedim. Quantifizierung Wartungsaufwände		Analyse in 6.5.8.5, **FF2.1**
Schwerpunkte	Prozesscharakteristika die Fernwartung limitieren	
1. Antw. 1:35:45	Die Verfügbarkeit von Informationen in den House-Keeping-Daten (Standard und erweitert), der fehlende Hardwarezugriff, die Verfügbarkeit des Teams zum entsprechenden Zeitpunkt und die Kontaktierungsmöglichkeit zum Satelliten sind die wesentlichen Probleme.	
2. Antw. 0:51:50	Es sind die Herausforderungen die sich generell aus der Fernwartung ergeben. Die Sichtbarkeit von offensichtlichen Problemen (zum Beispiel Überhitzungen) sind nur messbar, wenn entsprechende Sensoren verbaut sind.	

Tabelle D.17.: Antworten zu Frage 8 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 8
Analysedim. Quantifizierung Wartungsaufwände		Analyse in 6.5.8.5, **FF2.1**
Schwerpunkte	Höhe des Personalaufwands	
1. Antw. 01:37:52 und 01:39:40	Es gibt keine 24h-Bereitsschaft beim Wartungspersonal. Die Anzahl der Personen kann schnell ansteigen, wenn ganze Subsysteme betroffen sind.	
2. Antw. 0:53:30	Durch die sehr aufwendige Prozedur der Entwicklungs- und Testphase ist ein Software-Update sehr zeit- und personalintensiv. Es müssen vor allem die Experten mit dem notwendigen Wissen verfügbar sein. Beim TET-1 gab es eine Wartungsphase, die fast ein halbes Jahr lang andauerte.	

Tabelle D.18.: Antworten zu Frage 9 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 9
Analysedim. Quantifizierung Wartungsaufwände		Analyse in 6.5.8.6, **FF2.2**
Schwerpunkte	Limitierungen durch unterbrechnungsbehaftete Kommunikation	
1. Antw. 01:38:30 und 01:44:30	Im normalen Betrieb sind nur zwei Kontakte pro Tag möglich. Fehler können dadurch teilweise erst 12 Stunden später entdeckt werden. Es muss einen Fallback geben, der in der Zwischenzeit einen sicheren Betrieb sicherstellt. Der Uplink beschränkt die Wartung nicht im allgemeinen, da ein Upload auf mehrere Orbits verteilt werden kann.	
2. Antw. 0:54:20	Die Kapazitäten zum Upload sind beschränkt. Es wäre möglich zusätzlichen (kommerzielle) Bodenstationen hinzuzunehmen, dies ist aber nur in der LEOP üblich.	

Tabelle D.19.: Antworten zu Frage 10 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 10
Analysedim. Quantifizierung Wartungsaufwände		Analyse in 6.5.8.7, **FF2.3**
Schwerpunkte	Limitierungen bei Systemzugriff via Telemetrie beziehungsweise Telekommandierung	
1. Antw. 1:37:20	Man kann keine zusätzlichen Messungen an der Hardware durchführen. Ausschließlich die integrierten Bauteile zur Diagnose (zum Beispiel zur Spannungsmessung) können ausgewertet werden.	
2. Antw. 0:55:40	Schnittstellen zum Einspielen von Software-Updates sind vorgegeben und können nicht umgangen werden. Beispiel: Ein direktes Flashen eines Chips in einem Subsystem ist nicht möglich, sondern nur über den zentralen Rechner. Beim TET-1 musst zum Beispiel die Software des Hauptrechners für ein Firmware-Updates des Reaktionsradsystems modifiziert werden.	

Tabelle D.20.: Antworten zu Frage 11 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 11
Analysedim. Modellannahmen		Analyse in 6.5.8.3, **FF1.3**
Schwerpunkte	Vorhersagemodell im Wartungsprozess	
1. Antw. 1:49:00	Grundsätzlich wird Hardware-basiert getestet, Softwaremodelle werden nur für Fehler genutzt, die nicht auf Hardware provozierbar sind.	
2. Antw. 1:02:50	Hardware-Modelle sind verfügbar um den Betrieb der Baugruppen in Software zu simulieren.	

Tabelle D.21.: Antworten zu Frage 12 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 12
Analysedim. Modellannahmen		Analyse in 6.5.8.10, **FF3.4**
Schwerpunkte	Sichtbarkeit von Änderungen auf Architekturebene	
1. Antw.	(Bekannt aus dem vorherigen Gespräch: Ist nicht möglich.)	
2. Antw. 1:04:00	Eine Verwendung [von Architekturen] für diesen Anwendungsfall ist nicht bekannt.	

Tabelle D.22.: Antworten zu Frage 13 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 13
Analysedim. Modellannahmen		Analyse in 6.5.8.1, **FF1.1**
Schwerpunkte	Modellbasierte Erprobungen von Konfigurationsänderungen	
1. Antw.	(Bekannt aus dem vorherigen Gespräch: Wird nicht durchgeführt.)	
2. Antw. 1:06:30	Zur Unterstützung der Entscheidungsträger ist eine abstrakte Argumentation für eine Rekonfiguration sehr nützlich. Je nach Mentalität und Expertise werden unterschiedliche Lösungen vorgeschlagen, dort bietet sich eine objektive Abwägung an.	

Tabelle D.23.: Antworten zu Frage 14 aus Ablauf des Wartungsprozesses

Kat. Ablauf des Wartungsprozesses		Frage 14
Analysedim. Modellannahmen		Analyse in 6.5.8.1, **FF1.1**
Schwerpunkte	Modelle für Ausführungskontext des Systems	
1. Antw.	(Information aus vorherigen Fragen verfügbar: Umwelteinflüsse können optional auch in Modellen nachgebildet und in der Simulation berücksichtig werden.)	
2. Antw. 1:09:30	Umweltmodelle sind vorhanden, aber jegliche Änderungen werden generell auf dem Teststand erprobt und bemessen.	

Tabelle D.24.: Antworten zu Frage 1 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 1
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Relevanz von kritischen Ausfällen	
1. Antw. 1:53:20	Die Experimente des TET-1 sind an einen engen Zeitraum gebunden, der grundsätzlich vom System auch die volle Leistung verlangt. Kleinere Puffer sind vorgesehen, aber nicht allzu viel. Ausfälle im produktiven Betrieb sind daher auch bei TET-1 kritisch.	
2. Antw. 1:13:20	(War nicht beteiligt, aber Beschreibung des Normalfalls in der Missionsplanung.) Es gibt einen Routinetest jeder Nutzlast nach dem Start. Für jedes Experiment gibt es einen fixen Zeitplan, der eingehalten werden muss. Zusätzlich gibt es teilweise Einschränkungen zu welcher Tageszeit ein Experiment durchgeführt werden kann. In der Missionszeit werden zeitliche Rücklagen vorgesehen.	

Tabelle D.25.: Antworten zu Frage 2 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 2
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Nutzung des Safe-Modes während Updates	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw.		

Tabelle D.26.: Antworten zu Frage 3 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 3
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Safe-Modus im langfristigen Betrieb	
1. Antw. 1:55:20	Nicht als langfristige Alternative, sondern nur zu Überbrückung von Kontaktaufnahmen oder Wartungsphasen.	
2. Antw. 1:15:45	Der Safe-Mode ist ein schonender gesunder Modus, der zugleich aber auch keinen produktiven Betrieb ermöglicht.	

Tabelle D.27.: Antworten zu Frage 4 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 4
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Anpassung des Safe-Modus seit Inbetriebnahme	
1. Antw.	(Bekannt aus dem vorherigen Gespräch: Eine Anpassung des Safe-Mode zum alleinigen Betrieb der Magnetspulen zur Not-Ausrichtung hat stattgefunden.)	
2. Antw. 1:16:30	Es gab eine Anpassung im ACS-Safe-Mode. Der globale Satelliten-Safe-Mode wurde nicht verändert.	

Tabelle D.28.: Antworten zu Frage 5 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 5
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Akzeptanz einer autonomen Anpassung mit stabiler Systemqualität und gleichbleibender Missionserfüllung	
1. Antw. 1:58:00	Die Akzeptanz wird hoch sein.	
2. Antw. 1:19:20	Kommt auf die Mission an, aber grundsätzlich ja.	

Tabelle D.29.: Antworten zu Frage 6 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 6
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Akzeptanz einer autonomen Anpassung mit sinkender Systemqualität und eingeschränkter Missionserfüllung	
1. Antw. 1:58:50	Persönliche Einschätzung: Ein Ingenieur würde sich über den fortlaufenden Betrieb grundsätzlich freuen anstatt den unproduktiven Safe-Mode nutzen zu müssen.	
2. Antw. 1:19:52	Wenn keine andere Alternative gefunden werden konnte, dann trifft dies zu. In Grenzfällen geht es um wenige Minuten um weitere Daten zu erfassen. Ein System im Safe-Mode liefert keine produktiven Daten, da sind Daten mit minderer Qualität noch besser.	

Tabelle D.30.: Antworten zu Frage 7 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 7
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Entscheidungsunterstützung ohne autonome Anpassung	
1. Antw. 2:00:10	Dies ist insbesondere für Entscheidungsträger oder Kunden wertvoll.	
2. Antw. 1:21:50	Die vorhandene Autonomie im TET-1 bezieht sich immer auf Schutzmechanismen. Es geht daher nicht darum Missionsdaten zu retten. Die Autonomie wurde gesteigert, um die Wartung zu vereinfachen.	

Tabelle D.31.: Antworten zu Frage 8 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 8
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.10, **FF3.4**
Schwerpunkte	Dokumentation des Rekonfigurationsprozesses	
1. Antw.	Bei steigender Anzahl von möglichen Konfigurationen ist eine Information was passiert ist sinnvoll. Bei geringer Anzahl nicht, da die Strategien bereits bekannt sind.	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.32.: Antworten zu Frage 9 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 9
Analysedim. Akzeptanz des Ansatzes		Analyse irrelevant
Schwerpunkte	Relevante Informationen bei der Änderung einer Konfiguration	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.33.: Antworten zu Frage 10 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 10
Analysedim. Prozessverständnis		Analyse in 6.5.8.7, **FF2.3**
Schwerpunkte	Autonomer Wechsel in den Safe-Modus	
1. Antw. 2:02:30	Der Surveillance-Monitor aggregiert die Informationen und leitet den Wechsel ein.	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.34.: Antworten zu Frage 11 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 11
Analysedim. Prozessverständnis		Analyse in 6.5.8.7, **FF2.3**
Schwerpunkte	Weitere autonome Moduswechsel	
1. Antw. 2:03:50	Standardgemäß werden alle Moduswechsel kommandiert. Grundsätzlich wird nur der Safe-Mode im Fehlerfall autonom angefahren. Zusätzlich wird beim Verlust der inertialen Orientierung in den Modus zur groben Sonnenausrichtung gewechselt.	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.35.: Antworten zu Frage 12 aus Autonomie und Transparenz im Wartungsprozess

Kat. Autonomie und Transparenz im Wartungsprozess		Frage 12
Analysedim. Prozessverständnis		Analyse irrelevant
Schwerpunkte	Erweiterungen der Systemautonomie seit Inbetriebnahme	
1. Antw. 2:04:45	Neben der Anpassung des Safe-Mode zum alleinigen Betrieb der Magnetspulen zur Not-Ausrichtung gab es keine umfangreichen Anpassungen.	
2. Antw. 1:23:40	Kleinere Anpassungen um den personellen Aufwand zu verringern, zum Beispiel automatische Power-Cycles wenn Folgen abschätzbar.	

Tabelle D.36.: Antworten zu Frage 1 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 1
Analysedim. Aufbau Entwurfsraum		Analyse in 6.5.8.8, **FF2.4**
Schwerpunkte	Beziehungen zwischen Architekturentwurf und produktiver Implementierung	
1. Antw. 2:11:50	Architekturen über Beziehungen zwischen Ressourcen sind nicht zentral, sondern die Mission steht im Vordergrund. Die Missionsanforderungen beschreiben wiederrum, welche Hardware benötigt wird. Entstehen Änderungen werden diese nur optional in der Architektur nachgetragen.	
2. Antw. 1:28:00	Architekturen sind durch die Strukturierung des Codes ableitbar, werden aber nicht kontinuierlich erzeugt.	

Tabelle D.37.: Antworten zu Frage 2 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 2
Analysedim. Akzeptanz des Ansatzes		Analyse irrelevant
Schwerpunkte	Einschätzung der Architektur als zentrales Element im Entwicklungsprozess	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw. 1:29:20	Frage abgelehnt, da wenig Konfidenz auf dem Feld.	

Tabelle D.38.: Antworten zu Frage 3 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 3
Analysedim. Aufbau Entwurfsraum		Analyse irrelevant
Schwerpunkte	Sichtbarkeit von Systemvariationen auf Architekturebene	
1. Antw. 2:14:10	Ist auf ACS-Ebene im C-Code bekannt, aber nur als Verzweigung im Code, nicht als sichtbare Objekte.	
2. Antw. 1:28:35	Abbildung von Konfigurationen durch Verzweigungsstrukturen im Code.	

Tabelle D.39.: Antworten zu Frage 4 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 4
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.11, **FF4.2**
Schwerpunkte	Nutzung von Informationen aus dem Architekturentwurf zur Analyse in der Wartung	
1. Antw. 2:05:50	Für die Entwicklung ist das sehr gut möglich, wenn zum Beispiel überprüft werden muss ob bestimmte Qualitäten auch nach diversen Ausfällen noch möglich sind. Weiterhin kann auch ein übertriebenes Redundanzdesign identifiziert werden.	
2. Antw.	(Frage nicht gestellt, da wenig Konfidenz auf dem Feld.)	

Tabelle D.40.: Antworten zu Frage 5 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 5
Analysedim. Akzeptanz des Ansatzes		Analyse irrelevant
Schwerpunkte	Nachverfolgbarkeit von Konfigurationen auf Architekturebene	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw.	(Frage nicht gestellt, da wenig Konfidenz auf dem Feld.)	

Tabelle D.41.: Antworten zu Frage 6 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 6
Analysedim. Aufbau Entwurfsraum		Analyse irrelevant
Schwerpunkte	Abwägung von konkreten Konfigurationsalternativen im Entwurf	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.42.: Antworten zu Frage 7 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 7
Analysedim. Akzeptanz des Ansatzes		Analyse irrelevant
Schwerpunkte	Relevanz der Fehlererkennung und -behandlung im Entwurf	
1. Antw.	(Frage aus Zeitgründen übersprungen.)	
2. Antw.	(Frage aus Zeitgründen übersprungen.)	

Tabelle D.43.: Antworten zu Frage 8 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 8
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Akzeptanz eines Mehraufwand zur Konfigurationsraumanalyse im Entwurf	
1. Antw. 01:41:55 und 02:17:40	Eine Akzeptanz hängt stark von der Integration in den Entwicklungsprozess ab. Einerseits ist eine schnelle Reaktion des Systems angestrebt, solange alles sachgemäß reagiert. Anderseits ist die Domäne sehr konservativ und jede Veränderung sollte zunächst begutachtet werden. Als Teil der Fehleranalyse ist der Ansatz integrierbar.	
2. Antw. 1:40:30	Unterscheidung wichtig zwischen kommerziellen und Einzelmission dabei wichtig. Eine Argumentation über den Zugewinn an Ausfallsicherheit möglich. Bei kommerziellen Missionen muss das System zudem schnell ins Weltall kommen.	

Tabelle D.44.: Antworten zu Frage 9 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 9
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.4, **FF1.4**
Schwerpunkte	Verfügbarkeit von Daten für qualitativen Prognosen	
1. Antw. 02:07:00 und 02:18:15	Die Berechnung von Kombinationen von Ressourcen ist mathematisch möglich, solange die Einzeldaten vorliegen. Die Missionsanforderungen lassen sich auf konkrete qualitative Anforderungen reduzieren. Für eine Steigerung der Voraussagequalität sind aber zusätzliche Simulationen von Störungen notwendig.	
2. Antw. 1:36:30	Mit steigendem Detaillierungsgrad des Entwurfs werden zusätzliche Analysen und Simulationen notwendig, zu Beginn basiert aber alles auf Datenblättern und Wissen aus vorherigen Missionen. Die Datenblätter flugerprobter Hardware sind sehr präzise, einige Details müssen aber von den Herstellern nachgefordert werden. Unschärfen der Daten (zum Beispiel durch Umwelteinflüsse) werden im Entwurf durch relativ hohe Schwellwerte abgefangen.	

Tabelle D.45.: Antworten zu Frage 10 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 10
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.4, **FF1.4**
Schwerpunkte	Akzeptanz der Unschärfe von Meta-Heuristiken in der Wartung	
1. Antw. 2:19:10	Solang die Lösungen nachvollziehbar oder dicht an den bestehenden Konzepten sind, ist mit Akzeptanz zu rechnen. Der Ansatz kann zur Ideenfindung unterstützen, die Entscheidungen sollten die Ingenieure weiterhin selber treffen.	
2. Antw. 1:42:20	Wenn das System nur Vorschläge macht, sollte die Akzeptanz hoch sein solange eine nachvollziehbare Begründung für Konfigurationswechsel geliefert wird.	

Tabelle D.46.: Antworten zu Frage 11 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 11
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.8, **FF2.4**
Schwerpunkte	Integration einer architekturbasierten Dokumentation in den Prozess	
1. Antw. 2:16:10	Bereitschaft einer architekturbezogenen Entwicklung von Systemanpassungen wäre eine gute Vorgabe für den Entwickler. Die Erfahrung der Entwickler [im Software Engineering] spielt dort mit rein.	
2. Antw.	(Frage nicht gestellt, da wenig Konfidenz auf dem Feld.)	

Tabelle D.47.: Antworten zu Frage 12 aus Anwendbarkeit des Ansatzes

Kat. Anwendbarkeit des Ansatzes		Frage 12
Analysedim. Akzeptanz des Ansatzes		Analyse in 6.5.8.9, **FF3.3**
Schwerpunkte	Abwägung der Vor- und Nachteile bei Anwendung des Ansatzes	
1. Antw. 01:43:20 und 02:20:45	Grundsätzliche Zustimmung als Ansatz zur Exploration von Alternativen. Viele Lösungen können dennoch durch normales Engineering ermittelt und gelöst werden, da immer eine Vielzahl von Fehlerkombinationen im Vorfeld untersucht werden. Eine Lösungsalternativen bleiben aber vermutlich unberücksichtigt. Gerade für frühe Phasen ist der Ansatz sehr gut geeignet. Vor der Inbetriebnahme würden diverse Phasen zur Qualitätssicherung allerdings weiterhin anfallen.	
2. Antw. 1:44:00	Vorteile sind, dass Experten unbefangener Alternativen abwägen und dass das Expertenwissen in einer Datenbasis festgehalten wird. Sollte der Ansatz über die Beratungsfunktion hinausgehen und zur Autonomie einsetzbar sein, wäre er sehr wertvoll. Nachteile sind, dass das System im Betrieb mit weiteren Informationen gespeist werden müsste um Prognosen laufen zu präzisieren, da Datenblätter die Realität nur in Teilen abbilden. Es gibt zum Beispiel beim Rauschverhalten Durchschnittswerte und Maximalwerte die stark kontextabhängig sind. Das Wissen der Experten muss integriert werden, damit oberflächlich schlechte Lösungen ausgeschlossen werden können und das historische Wissen genutzt wird. Weiterhin muss das Projektmanagement bereit sein zusätzliche Arbeit in die Erprobung des Ansatzes zu investieren. Dies ist in einer hochspezialisierten Domäne schwierig.	

Literaturverzeichnis

[AAC14] Andrade, Hugo S. ; Almeida, Eduardo ; Crnkovic, Ivica: Architectural bad smells in software product lines: An exploratory study. In: *Proceedings of the 11th Working IEEE/IFIP Conference on Software Architecture* ACM, 2014

[ABB+02] Atkinson, Colin ; Bayer, Joachim ; Bunse, Christian ; Kamsties, Erik ; Laitenberger, Oliver ; Laqua, Roland ; Muthig, Dirk ; Paech, Barbara ; Wüst, Jürgen ; Zettel, Jörg: *Component-based Product Line Engineering with UML*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2002. – ISBN 0–201–73791–4

[ABG+13a] Aleti, Aldeida ; Buhnova, Barbora ; Grunske, Lars ; Koziolek, Anne ; Meedeniya, Indika: Software Architecture Optimization Methods: A Systematic Literature Review. In: *IEEE Transactions on Software Engineering* 39 (2013), May, Nr. 5, S. 658–683. – ISSN 0098–5589

[ABG+13b] Aleti, Aldeita ; Buhnova, Barbora ; Grunske, Lars ; Koziolek, Anne ; Meedeniya, Indika: Software Architecture Optimization Methods: A Systematic Literature Review. In: *IEEE Transactions on Software Engineering* 39 (2013), Dezember, Nr. 5, S. 658–683. http://dx.doi.org/10.1109/TSE.2012.64. – DOI 10.1109/TSE.2012.64

[ABGM09] Aleti, A ; Bjornander, S. ; Grunske, Lars ; Meedeniya, I: ArcheOpterix: An extendable tool for architecture optimization of AADL models. In: *Proceedings of the ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, 2009, S. 61–71

[AMS+10] Axmann, Robert ; Mühlbauer, Peter ; Spörl, Andreas ; Turk, Michael ; Föckersperger, Stefan ; Schmolke, Jürgen: Operations Concept And Challenges For 11 Different Payloads On The TET-1 Mini-Satellite. In: *Proceedings of the 2010 4S Symposium - Small Satellite Systems and Services*, 2010

[Avi67] Avižienis, Algirdas: Design of Fault-tolerant Computers. In: *Proceedings of 1967 Fall Joint Computer Conference*, ACM, 1967, S. 733–743

[Avi97] Avizienis, Algirdas: Toward Systematic Design of Fault-tolerant Systems. In: *IEEE Computer* 30 (1997), Nr. 4, S. 51–58. http://dx.doi.org/10.1109/2.585154. – DOI 10.1109/2.585154. – ISSN 0018–9162

[Bav50] Bavelas, Alex: Communication Patterns in Task-Oriented Groups. In: *Acoustical Society of America* 22 (1950), November, Nr. 6, S. 725–730. http://dx.doi.org/10.1121/1.1906679. – DOI 10.1121/1.1906679

[BBN10] Barley, Bryan ; Bacskay, Allen ; Newhouse, Marilyn: Heritage and Advanced Technology Systems Engineering - Lessons Learned from NASA Deep Space Missions. In: *Proceedings of the AIAA SPACE 2010 Conference & Exposition*, 2010

[BHS07] Buschmann, Frank ; Henney, Kelvin ; Schimdt, Douglas: *Pattern-oriented Software Architecture: on patterns and pattern language*. Bd. 5. Wiley, 2007. – ISBN 978–0–470–05902–9

[BKR09] Becker, Steffen ; Koziolek, Heiko ; Reussner, Ralf: The Palladio Component Model for Model-driven Performance Prediction. In: *Systems & Software* 82 (2009), Januar, Nr. 1, S. 3–22. http://dx.doi.org/10.1016/j.jss.2008.03.066. – DOI 10.1016/j.jss.2008.03.066. – ISSN 0164–1212

[BLLS14] Baller, Hauke ; Lity, Sascha ; Lochau, Malte ; Schaefer, Ina: Multi-objective test suite optimization for incremental product family testing. In: *Proceedings of 7th International Conference on Software Testing, Verification and Validation*, 2014, S. 303–312

[BPG13] Barnes, Jeffrey M. ; Pandey, Ashutosh ; Garlan, David: Automated Planning for Software Architecture Evolution. In: *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. Piscataway, NJ, USA : IEEE Press, 2013. – ISBN 978–1–4799–0215–6, S. 213–223

[BSB08] Bommer, Christoph ; Spindler, Markus ; Barr, Volker: *Softwarewartung - Grundlagen, Management und Wartungstechniken*. dpunkt Verlag, 2008. – ISBN 978–3–89864–482–2

[BSRC10] Benavides, David ; Segura, Sergio ; Ruiz-Cortés, Antonio: Automated analysis of feature models 20 years later: A literature review. In: *Information Systems* 35 (2010), Nr. 6, S. 615–636. http://dx.doi.org/10.1016/j.is.2010.01.001. – DOI 10.1016/j.is.2010.01.001

[BW84] Basili, V. R. ; Weiss, D. M.: A Methodology for Collecting Valid Software Engineering Data. In: *IEEE Transactions on Software Engineering* SE-10 (1984), November, Nr. 6, S. 728–738. http://dx.doi.org/10.1109/TSE.1984.5010301. – DOI 10.1109/TSE.1984.5010301. – ISSN 0098–5589

[CA05] Czarnecki, Krzysztof ; Antkiewicz, Michał: Mapping features to models: A template approach based on superimposed variants. In: *Proceedings of the International Conference on Generative Programming and Component Engineering* Springer-Verlag, 2005, S. 422–437

[CCG+02] Cimatti, Alessandro ; Clarke, Edmund ; Giunchiglia, Enrico ; Giunchiglia, Fausto ; Pistore, Marco ; Roveri, Marco ; Sebastiani, Roberto ; Tacchella, Armando: Nusmv 2: An opensource tool for symbolic model checking. In: *Proceedings of the 14th International Conference on Computer Aided Verification*, 2002, S. 359–364

[CDL15] Chien, Jay-Syin W. ; Duphily, Roland J. ; Lee, Yum T.: The reliability, availability, and maintainability of ground systems in support of space missions. In: *Proceedings of the 2015 Annual Reliability and Maintainability Symposium*, 2015. – ISSN 0149–144X, S. 1–6

[CDPEV08] Canfora, Gerardo ; Di Penta, Massimiliano ; Esposito, Raffaele ; Villani, Maria L.: A Framework for QoS-aware Binding and Re-binding of Composite Web Services. In: *Systems and Software* 81 (2008), Oktober, Nr. 10, S. 1754–1769. http://dx.doi.org/10.1016/j.jss.2007.12.792. – DOI 10.1016/j.jss.2007.12.792. – ISSN 0164–1212

[CE00] Czarnecki, Krzysztof ; Eisenecker, Ulrich W.: *Generative programming: methods, tools, and applications*. Bd. 16. Addison-Wesley Professional, 2000. – ISBN 978–0201309775

[CHS11] Clarke, Dave ; Helvensteijn, Michiel ; Schaefer, Ina: Abstract delta modeling. In: *ACM Sigplan Notices* 46 (2011), Nr. 2, S. 13–22. http://dx.doi.org/10.1145/1942788.1868298. – DOI 10.1145/1942788.1868298

[CJCG08] Cieslewski, G. ; Jacobs, A. ; Conger, C. ; George, A.: Advanced Space Computing with System-Level Fault Tolerance (Invited Talk). In: *Proceedings of the 1st Workshop on Fault-Tolerant Spaceborne Computing – Employing New Technologies*, 2008

[CPNMH18] Chemweno, Peter ; Peter Nganga Muchiri, Liliane P. ; Horenbeek, Adriaan V.: Risk assessment methodologies in maintenance decision making: A review of dependability modelling approaches. In: *Reliability Engineering & System Safety* 173 (2018), Mai, S. 64–77. – ISSN 0951–8320

[CSW+15] Cacilo, Andrej ; Schmidt, Sarah ; Wittlinger, Philipp ; Herrmann, Florian ; Bauer, Wilhelm ; Sawade, Oliver ; Doderer, Hannes ; Hartwig, Matthias ; Scholz, Volker: Hochautomatisiertes Fahren auf Autobahnen – Industriepolitische Schlussfolgerungen / Fraunhofer-Institut für Arbeitswirtschaft und Organisation. Version: 2015. http://publica.fraunhofer.de/dokumente/N-372308.html. 2015. – Forschungsbericht. – Dienstleistungsprojekt 15/14

[Czi15] Czichos, Horst: *Mechatronik-Grundlagen und Anwendungen technischer Systeme*. 3. Springer Vieweg, 2015. – ISBN 978–3–658–09950–3

[Die12] Diestel, Reinhard: *Graduate Texts in Mathematics*. Bd. 173: *Graph Theory*. 4. Springer-Verlag, 2012. – ISBN 978–3–642–14278–9

[DL94] Dugan, Joanne B. ; Lyu, Michael R.: System reliability analysis of an N-version programming application. In: *IEEE Transactions on Reliability* 43 (1994), Dec, Nr. 4, S. 513–519. http://dx.doi.org/10.1109/24.370232. – DOI 10.1109/24.370232. – ISSN 0018–9529

[DN02] Dobrica, Liliana ; Niemelä, Eila: A survey on software architecture analysis methods. In: *IEEE Transactions on Software Engineering* 28 (2002), Jul, Nr. 7, S. 638–653. http://dx.doi.org/10.1109/TSE.2002.1019479. – DOI 10.1109/TSE.2002.1019479. – ISSN 0098–5589

[DPAM02] Deb, K. ; Pratap, A. ; Agarwal, S. ; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. In: *IEEE Transactions on Evolutionary Computation* 6 (2002), April, Nr. 2, S. 182–197. http://dx.doi.org/10.1109/4235.996017. – DOI 10.1109/4235.996017. – ISSN 1089–778X

[EBG12] Ensan, Faezeh ; Bagheri, Ebrahim ; Gašević, Dragan: Evolutionary Search-based Test Generation for Software Product Line Feature Models. In: *Proceedings of the 24th International Conference on Advanced Information Systems Engineering*. Berlin, Heidelberg : Springer-Verlag, 2012. – ISBN 978–3–642–31094–2, S. 613–628

[EKN+12] Esteve, Marie-Aude ; Katoen, Joost-Pieter ; Nguyen, Viet Y. ; Postma, Bart ; Yushtein, Yuri: Formal correctness, safety, dependability, and performance analysis of a satellite. In: *Proceedings of the 34th International Conference on Software Engineering*, 2012, S. 1022–1031

[EMKS+17] Evans, David ; Martinez, José ; Korte-Stapff, Moritz ; Brighenti, Attilio ; Brighenti, Chiara ; Biancat, Jacopo: Data Mining to Drastically Improve Spacecraft Telemetry Checking. Version: 2017. http://dx.doi.org/10.1007/978-3-319-51941-8_4. In: *Space Operations: Contributions from the Global Community*. Springer-Verlag, 2017. – DOI 10.1007/978–3–319–51941–8_4, S. 87–113

[Eur09] European Cooperation for Space Standardization: *ECSS-E-ST-40C Space Engineering - Software Standard*. 2009 http://ecss.nl/standard/ecss-e-st-40c-software-general-requirements/

[Eur13] European Cooperation for Space Standardization: *ECSS-E-HB-40A Space Engineering – Software Engineering Handbook*. 2013 http://ecss.nl/ecss-handbooks/

[FBB+99] Fowler, Martin ; Beck, Kent ; Brant, John ; Opdyke, William ; Roberts, Don: *Refactoring: Improving the design of existing code*. Addison-Wesley Professional, 1999. – ISBN 978–0201485677

[FFH13] Frey, Sören ; Fittkau, Florian ; Hasselbring, Wilhelm: Search-based Genetic Optimization for Deployment and Reconfiguration of Software in the Cloud. In: *Proceedings of the 35th International Conference on Software Engineering*, 2013, S. 512–521

[FGH06] Feiler, Peter H. ; Gluch, David P. ; Hudak, John J.: The architecture analysis & design language (AADL): An introduction / Carnegie-Mellon University Pittsburgh PA Software Engineering Institute. Version: 2006. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=7879. 2006. – Forschungsbericht

[FH06] Florentz, Bastian ; Huhn, Michaela: Embedded Systems Architecture: Evaluation and Analysis. In: Hofmeister, Christine (Hrsg.) ; Crnkovic, Ivica (Hrsg.) ; Reussner, Ralf (Hrsg.): *Quality of Software Architectures* Bd. 4214. Springer Berlin / Heidelberg, 2006. – ISBN 978–3–540–48819–4, S. 145–162

[FL03] Fox, Maria ; Long, Derek: PDDL2. 1: An extension to PDDL for expressing temporal planning domains. In: *Artificial Intelligence Research* (2003). http://dx.doi.org/10.1613/jair.1129. – DOI 10.1613/jair.1129

[FLK+10] Föckersperger, Stefan ; Lattner, Klaus ; Kaiser, Clemens ; Eckert, Silke ; Ritzmann, Swen ; Axmann, Robert ; Turk, Michael: *The On-Orbit Verification Mission TET-1: Project Status of the Small Satellite Mission and Outlook for a One Year Mission Operation Phase*. http://www.kayser-threde.de/tet/downloads/Papers/4S_Paper_TET-1_Madeira-2010.pdf. Version: 2010

[Fre77] Freeman, Linton C.: A Set of Measures of Centrality Based on Betweenness. In: *Sociometry* 40 (1977), März, Nr. 1, S. 35–41. http://dx.doi.org/10.2307/3033543. – DOI 10.2307/3033543. – ISSN 00380431

[GBSC09] Garlan, David ; Barnes, Jeffrey M. ; Schmerl, Bradley ; Celiku, Orieta: Evolution styles: Foundations and tool support for software architecture evolution. In: *Proceedings of the 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 2009, S. 131–140

[GC03] Ganek, Alan G. ; Corbi, T. A.: The Dawning of the Autonomic Computing Era. In: *IBM Systems* 42 (2003), Januar, Nr. 1, S. 5–18. http://dx.doi.org/10.1147/sj.421.0005. – DOI 10.1147/sj.421.0005. – ISSN 0018–8670

[GCD10] Garvin, Brady J. ; Cohen, Myra B. ; Dwyer, Matthew B.: Evaluating improvements to a meta-heuristic search for constrained interaction testing. In: *Empirical Software Engineering* 16 (2010), Nr. 1, S. 61–102. – ISSN 1573–7616

[GKKM10] Goltz, Ursula ; Khakpour, Narges ; Knieke, Christoph ; Märtin, Lukas: Behavioral Modeling of IT Ecosystems / TU Braunschweig. Version: 2010. https://www.tu-braunschweig.de/Medien-DB/ips/techrep/modelling-it-eco-sys.pdf. 2010. (NTH School for IT Ecosystems Reports). – Forschungsbericht

[GL10] Gläser, Jochen ; Laudel, Grit: *Experteninterviews und qualitative Inhaltsanalyse als Instrumente rekonstruierender Untersuchungen*. 4. Wiesbaden : VS Verlag für Sozialwissenschaften, 2010. – ISBN 978–3–531–17238–5

[GLB+07] Grunske, Lars ; Lindsay, Peter ; Bondarev, Egor ; Papadopoulos, Yiannis ; Parker, David: An Outline of an Architecture-Based Method for Optimizing Dependability Attributes of Software-Intensive Systems. Version: 2007. http://dx.doi.org/10.1007/978-3-540-74035-3_9. In: Lemos, Rogério de (Hrsg.) ; Gacek, Cristina (Hrsg.) ; Romanovsky, Alexander (Hrsg.): *Architecting Dependable Systems IV* Bd. 4615. Springer-Verlag, 2007. – DOI 10.1007/978–3–540–74035–3_9, S. 188–209

[GPEM09] Garcia, Joshua ; Popescu, Daniel ; Edwards, George ; Medvidovic, Nenad: Toward a Catalogue of Architectural Bad Smells. In: *Proceedings of the 5th International Conference on the Quality of Software Architectures: Architectures for Adaptive Software Systems*. Berlin, Heidelberg : Springer-Verlag, 2009 (QoSA '09). – ISBN 978–3–642–02350–7, S. 146–162

[GRG+14] Goltz, Ursula ; Reussner, Ralf H. ; Goedicke, Michael ; Hasselbring, Wilhelm ; Märtin, Lukas ; Vogel-Heuser, Birgit: Design for future: managed software evolution - The DFG priority programme for long-living software systems. In: *Computer Science - Research and Development (Special Issue)* (2014), 10

[Gru07] Grunske, Lars: Early Quality Prediction of Component-based Systems - A Generic Framework. In: *Systems and Software* 80 (2007), Mai, Nr. 5, S. 678–686. http://dx.doi.org/10.1016/j.jss.2006.08.014. – DOI 10.1016/j.jss.2006.08.014. – ISSN 0164–1212

[HA05] Hove, Siw E. ; Anda, Bente: Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research. In: *Proceedings of the 11th IEEE International Software Metrics Symposium*, IEEE Computer Society, 2005. – ISBN 0–7695–2371–4, S. 10–23

[Hei14] Hein, Andreas M.: How to Assess Heritage Systems in the Early Phases? In: *Proceedings of the 6th International Conference on Systems & Concurrent Engineering for Space Applications*, 2014

[HHPS08] Hallsteinsen, Svein ; Hinchey, Mike ; Park, Sooyong ; Schmid, Klaus: Dynamic software product lines. In: *IEEE Computer* 41 (2008), April, Nr. 4. http://dx.doi.org/10.1109/MC.2008.123. – DOI 10.1109/MC.2008.123. – ISSN 0018–9162

[HJZ+17] Heinrich, Robert ; Jung, Reiner ; Zirkelbach, Christian ; Hasselbring, Wilhelm ; Reussner, Ralf: An Architectural Model-Based Approach to Quality-aware DevOps in Cloud Applications. In: Mistrik, Ivan (Hrsg.) ; Bahsoon, Rami (Hrsg.) ; Ali, Nour (Hrsg.) ; Heisel, Maritta (Hrsg.) ; Maxim, Bruce (Hrsg.): *Software Architecture for Big Data and the Cloud*. New York, NY, USA : Elsevier Science Inc., 2017. – ISBN 9780128054673

[Hol92] Holland, John H.: Genetic algorithms. In: *Scientific American* 267 (1992), Juli, Nr. 1, S. 66–73. http://dx.doi.org/10.1038/scientificamerican0792-66. – DOI 10.1038/scientificamerican0792–66

[HPP+13] Henard, Christopher ; Papadakis, Mike ; Perrouin, Gilles ; Klein, Jacques ; Traon, Yves L.: PLEDGE: A Product Line Editor and Test Generation Tool. In: *Proceedings of the 17th International Software Product Line Conference Co-located Workshops*, ACM, 2013. – ISBN 978–1–4503–2325–3, S. 126–129

[HRRS14] Haber, Arne ; Rendel, Holger ; Rumpe, Bernhard ; Schaefer, Ina: Delta Modeling for Software Architectures. In: *Computing Research Repository* abs/1409.2358 (2014). http://arxiv.org/abs/1409.2358

[HSH+14] Howell, Steve B. ; Sobeck, Charlie ; Haas, Michael ; Still, Martin ; Barclay, Thomas ; Mullally, Fergal ; Troeltzsch, John ; Aigrain, Suzanne ; Bryson, Stephen T. ; Caldwell, Doug u. a.: The K2 mission: characterization and early results. In: *Publications of the Astronomical Society of the Pacific* 126 (2014), Februar, Nr. 938, S. 398–408. http://dx.doi.org/10.1086/676406. – DOI 10.1086/676406

[ISO09] Norm ISO 22179 2009. *Intelligent transport systems – Full speed range adaptive cruise control (FSRA) systems – Performance requirements and test procedures*

[ISO10] Norm ISO 15622 2010. *Intelligent transport systems – Adaptive Cruise Control systems – Performance requirements and test procedures*

[JJH+08] Jung, Gueyoung ; Joshi, Kaustubh R. ; Hiltunen, Matti A. ; Schlichting, Richard D. ; Pu, Calton: Generating Adaptation Policies for Multi-tier Applications in Consolidated Server Environments. In: *Proceedings of the 2008 International Conference on Autonomic Computing*, 2008 (ICAC'08), S. 23–32

[Jon01] Jones, Willie D.: Keeping cars from crashing. In: *IEEE Spectrum* 38 (2001), Nr. 9, S. 40–45. – ISSN 0018–9235

[JSE96] Jones, Bryan F. ; Sthamer, H.-H. ; Eyres, David E.: Automatic structural testing using genetic algorithms. In: *Software Engineering Journal* 11 (1996), September, Nr. 5, S. 299–306. http://dx.doi.org/10.1049/sej.1996.0040. – DOI 10.1049/sej.1996.0040

[KABC96] Kazman, Rick ; Abowd, Gregory ; Bass, Len ; Clements, Paul: Scenario-based analysis of software architecture. In: *IEEE Software* 13 (1996), Nr. 6, S. 47–55. http://dx.doi.org/10.1109/52.542294. – DOI 10.1109/52.542294

[Kai14] Kaiser, Robert: *Qualitative Experteninterviews – Konzeptionelle Grundlagen und praktische Durchführung*. Springer-Verlag, 2014. http://dx.doi.org/10.1007/978-3-658-02479-6. http://dx.doi.org/10.1007/978-3-658-02479-6. – ISBN 978–3–658–02479–6

[KC03] Kephart, Jeffrey O. ; Chess, David M.: The Vision of Autonomic Computing. In: *IEEE Computer* 36 (2003), S. 41–50. http://dx.doi.org/10.1109/MC.2003.1160055. – DOI 10.1109/MC.2003.1160055

[KCH+90] Kang, Kyo C. ; Cohen, Sholom G. ; Hess, James A. ; Novak, William E. ; Peterson, A S.: Feature-oriented domain analysis (FODA) feasibility study / Carnegie-Mellon University Pittsburgh PA Software Engineering Inst. Version: 1990. https://resources.sei.cmu.edu/asset_files/TechnicalReport/1990_005_001_15872.pdf. 1990. – Forschungsbericht

[KDGR13] Konersmann, Marco ; Durdik, Zoya ; Goedicke, Michael ; Reussner, Ralf H.: Towards architecture-centric evolution of long-living systems (the ADVERT approach). In: *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures* ACM, 2013, S. 163–168

[KG14] Kang, Sungwon ; Garlan, David: Architecture-based planning of software evolution. In: *Software Engineering and Knowledge Engineering* 24 (2014), Nr. 02, S. 211–241. http://dx.doi.org/10.1142/S0218194014500090. – DOI 10.1142/S0218194014500090

[KKC00] Kazman, Rick ; Klein, Mark ; Clements, Paul: ATAM: Method for architecture evaluation / Carnegie-Mellon University Pittsburgh PA Software Engineering Instute. Version: 2000. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=513805. 2000. – Forschungsbericht

[KL86] Knight, John C. ; Leveson, Nancy G.: An experimental evaluation of the assumption of independence in multiversion programming. In: *IEEE Transactions on Software Engineering* (1986), Nr. 1, S. 96–109

[KM85] Kramer, Jeff ; Magee, Jeff: Dynamic configuration for distributed systems. In: *IEEE Transactions on Software Engineering* (1985), April, Nr. 4, S. 424–436. http://dx.doi.org/10.1109/TSE.1985.232231. – DOI 10.1109/TSE.1985.232231

[KM90] Kramer, Jeff ; Magee, Jeff: The evolving philosophers problem: Dynamic change management. In: *IEEE Transactions on Software Engineering* 16 (1990), Nr. 11, S. 1293–1306. http://dx.doi.org/10.1109/TSE.1985.232231. – DOI 10.1109/TSE.1985.232231

[KM07] Kramer, Jeff ; Magee, Jeff: Self-Managed Systems: An Architectural Challenge. In: *Proceedings of the 2017 Future of Software Engineering Symposium*, 2007, S. 259–268

[Koz11] Koziolek, Anne: *Automated Improvement of Software Architecture Models for Performance and other Quality Attributes*, Karlsruhe Institute of Technology, Diss., 2011

[KR11] Koziolek, Anne ; Reussner, Ralf: Towards a generic quality optimisation framework for component-based system models. In: *Proceedings of the 14th International ACM Sigsoft Symposium on Component based Software Engineering*, 2011, S. 103–108

[Kva96] Kvale, Steinar: *InterViews: An Introduction to Qualitative Research Interviewing*. SAGE Publications Inc., 1996. – ISBN 978–0–80395–819–7

[LEEC11] Li, Yan R. ; Etemaadi, Ramin ; Emmerich, Michael T. M. ; Chaudron, Michel R V.: An evolutionary multiobjective optimization approach to component-based software architecture design. In: *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*, 2011, S. 432–439

[LHLE15] Lopez-Herrejon, Roberto E. ; Linsbauer, Lukas ; Egyed, Alexander: A systematic mapping study of search-based software engineering for software product lines. In: *Information and software technology* 61 (2015), Mai, S. 33–51. http://dx.doi.org/10.1016/j.infsof.2015.01.008. – DOI 10.1016/j.infsof.2015.01.008

[LHSR12] Löw, Sebastian ; Herman, Jaap ; Schulze, Daniel ; Raschke, Christian: Modes and More – Finding the Right Attitude for TET-1. In: *Proceedings of the 12th International Conference on Space Operations*, 2012

[Lil16] Lilienthal, Carola: *Langlebige Software-Architekturen: Technische Schulden analysieren, begrenzen und abbauen.* dpunkt Verlag, 2016. – ISBN 9783864918834

[LLL+14] Lochau, Malte ; Lity, Sascha ; Lachmann, Remo ; Schaefer, Ina ; Goltz, Ursula: Delta-oriented model-based integration testing of large-scale systems. In: *Systems and Software* 91 (2014), S. 63–84. http://dx.doi.org/10.1016/j.jss.2013.11.1096. – DOI 10.1016/j.jss.2013.11.1096

[LLL+15] Lachmann, Remo ; Lity, Sascha ; Lischke, Sabrina ; Beddig, Simon ; Schulze, Sandro ; Schaefer, Ina: Delta-oriented test case prioritization for integration testing of software product lines. In: *Proceedings of the 19th International Conference on Software Product Line*, 2015, S. 81–90

[LST78] Lientz, Bennet P. ; Swanson, E B. ; Tompkins, Gail E.: Characteristics of application software maintenance. In: *Communications of the ACM* 21 (1978), Juni, Nr. 6, S. 466–471. http://dx.doi.org/10.1145/359511.359522. – DOI 10.1145/359511.359522

[Lyu95] Lyu, Michael R.: *Software Fault Tolerance.* New York, NY, USA : John Wiley & Sons, Inc., 1995. – ISBN 0471950688

[Mat09] Matevska, Jasminka: *Architekturbasierte erreichbarkeitsoptimierte Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit*, Department of Computer Science, University of Oldenburg, Oldenburg, Germany, Diss., 2009

[May13] Mayer, Horst O.: *Interview und schriftliche Befragung: Grundlagen und Methoden empirischer Sozialforschung.* Verlag Walter De Gruyter, 2013. – ISBN 9783486717624

[MC14] Markley, F. L. ; Crassidis, John L.: *Fundamentals of spacecraft attitude determination and control.* Bd. 33. Springer-Verlag, 2014. – ISBN 978–1–4939–5569–5

[MD08] Montenegro, Sergio ; Dittrich, Lutz: *The Core Avionics System for the DLR Compact-Satellite Series.* 2008

[MFKR18] Märtin, Lukas ; Forjahn, Nils-André ; Koziolek, Anne ; Reussner, Ralf H.: Guidance of Architectural Changes in Dependable Systems with Varying Operational Modes. In: *Proceedings of the 12th European Conference on Software Architecture*, 2018, S. 37–45

[MGLW15] Maurer, Markus ; Gerdes, J C. ; Lenz, Barbara ; Winner, Hermann: *Autonomes Fahren -Technische, rechtliche und gesellschaftliche Aspekte*. Springer-Verlag, 2015. – ISBN 978–3–662–45854–9

[MH07] Matevska, Jasminka ; Hasselbring, Wilhelm: A scenario-based approach to increasing service availability at runtime reconfiguration of component-based systems. In: *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications* IEEE, 2007, S. 137–148

[MKBR10] Martens, Anne ; Koziolek, Heiko ; Becker, Steffen ; Reussner, Ralf: Automatically Improve Software Architecture Models for Performance, Reliability, and Cost Using Evolutionary Algorithms. In: *Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering*. New York, NY, USA : ACM, 2010. – ISBN 978–1–60558–563–5, S. 105–116

[MKMG97] Monroe, Robert T. ; Kompanek, Andrew ; Melton, Ralph ; Garlan, David: Architectural styles, design patterns, and objects. In: *IEEE Software* 14 (1997), Nr. 1, S. 43–52. http://dx.doi.org/10.1109/52.566427. – DOI 10.1109/52.566427

[MKR15] Märtin, Lukas ; Koziolek, Anne ; Reussner, Ralf H.: Quality-oriented Decision Support for maintaining Architectures of fault-tolerant Space Systems. In: *Proceedings of the 2015 European Conference on Software Architecture Workshops*, 2015, S. 49:1–49:5

[MMMR12] Malek, Sam ; Medvidovic, Nenad ; Mikic-Rakic, Marija: An Extensible Framework for Improving a Distributed Software System's Deployment Architecture. In: *IEEE Transactions on Software Engineering* 38 (2012), Januar, Nr. 1, S. 73–100. http://dx.doi.org/10.1109/TSE.2011.3. – DOI 10.1109/TSE.2011.3

[MMT08] Maibaum, Olaf ; Montenegro, Sergio ; Terzibaschian, Thomas: Robustness as Key to Success for Space Missions. Version: 2008. http://dx.doi.org/10.1007/978-1-84800-261-6_11. In: Schuster, Alfons (Hrsg.): *Robust Intelligent Systems*. London : Springer-Verlag, 2008. – DOI 10.1007/978–1–84800–261–6_11. – ISBN 978–1–84800–261–6, S. 232–248

[MN14] Märtin, Lukas ; Nicolai, Anja: Towards Self-Reconfiguration of Space Systems on Architectural Level based on Qualitative Ratings. In: *Proceedings of the 35th IEEE International Aerospace Conference*, 2014

[MSH+13] Märtin, Lukas ; Schatalov, Maxim ; Hagner, Matthias ; Maibaum, Olaf ; Goltz, Ursula: A Methodology for Model-based Development and Automated Verification of Software for Aerospace Systems. In: *Proceedings of the 34th IEEE Aerospace Conference*, 2013

[NAS12] NASA (National Aeronautics and Space Administration): *NASA-HDBK-1002: Fault Management Handbook (Draft 2)*. NASA, 2012 http://www.nasa.gov/pdf/636372main_NASA-HDBK-1002_Draft.pdf

[NHO+11] Nothdurft, Tobias ; Hecker, Peter ; Ohl, Sebastian ; Saust, Falko ; Maurer, Markus ; Reschka, Andres ; Böhmer, Jürgen R.: Stadtpilot: First fully autonomous test drives in urban traffic. In: *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems*, 2011. – ISSN 2153–0009, S. 919–924

[Nol15] Noll, Thomas: Safety, Dependability and Performance Analysis of Aerospace Systems. In: Artho, Cyrille (Hrsg.) ; Ölveczky, Peter C. (Hrsg.): *Proceedings of the 3rd International Workshop on Formal Techniques for Safety-Critical Systems*, 2015, S. 17–31

[NS12] Naab, Matthias ; Stammel, Johannes: Architectural flexibility in a software-system's life-cycle: systematic construction and exploitation of flexibility. In: *Proceedings of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures*, 2012, S. 13–22

[Ohl14] Ohl, Sebastian: *Fusion von Umfeld wahrnehmenden Sensoren in städtischer Umgebung*. Shaker Verlag, 2014 (Berichte aus der Elektrotechnik). – ISBN 9783844030341

[Pan12] Pantaleoni, Mauro: XMM-Newton's operational challenge of changing the attitude control to 4 active reaction wheels, after 12 years of routine operations. Version: Juni 2012. `http://dx.doi.org/10.2514/6.2012-1275587`. In: *SpaceOps 2012*. 2012. – DOI 10.2514/6.2012–1275587

[Pat90] Patton, Michael Q.: *Qualitative Evaluation and Research Methods*. SAGE Publications, 1990. – ISBN 9780803937796

[PBDL05] Pohl, Klaus ; Böckle, Günter ; Der Linden, Frank J.: *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005. – ISBN 978–3540243724

[PC14] Pecchioli, Mauro ; Carranza, Juan M.: Highlights of the European Ground Systems-Common Core Initiative. In: *Proceedings of the SpaceOps 2014 Conference* AIAA, 2014, S. 219–241

[PHP99] Pargas, Roy P. ; Harrold, Mary J. ; Peck, Robert R.: Test-data generation using genetic algorithms. In: *Software Testing Verification and Reliability* 9 (1999), Dezember, Nr. 4, S. 263–282. `http://dx.doi.org/10.1.1.33.7219`. – DOI 10.1.1.33.7219

[PMS18] Phillips, Dewanne M. ; Mazzuchi, Thomas A. ; Sarkani, Shahram: An architecture, system engineering, and acquisition approach for space system software resiliency. In: *Information and Software Technology* 94 (2018), S. 150–164. `http://dx.doi.org/10.1016/j.infsof.2017.10.006`. – DOI 10.1016/j.infsof.2017.10.006. – ISSN 0950–5849

[QRV+15] Quinton, Clément ; Rabiser, Rick ; Vierhauser, Michael ; Grünbacher, Paul ; Baresi, Luciano: Evolution in Dynamic Software Product Lines: Challenges and Perspectives. In: *Proceedings of the 19th International Conference on Software Product Line*, ACM, 2015, S. 126–130

[RBB+11] Reussner, Ralf ; Becker, Steffen ; Burger, Erik ; Happe, Jens ; Hauck, Michael ; Koziolek, Anne ; Koziolek, Heiko ; Krogmann, Klaus ; Kuperberg, Michael: The Palladio Component Model / KIT, Fakultät für Informatik. 2011 (Karlsruhe Reports in Informatics 2011,14). – Forschungsbericht

[RBH+16] Reussner, Ralf H. ; Becker, Steffen ; Happe, Jens ; Heinrich, Robert ; Koziolek, Anne ; Koziolek, Heiko ; Kramer, Max ; Krogmann, Klaus: *Modeling and simulating software architectures: The palladio approach.* MIT Press, 2016. – ISBN 9780262034760

[RH08] Reussner, Ralf H. ; Hasselbring, Wilhelm: *Handbuch der Software-Architektur.* 2. Heidelberg : dpunkt Verlag, 2008. – ISBN 978–3–89864–559–1

[RHBR17] Rostami, Kiana ; Heinrich, Robert ; Busch, Axel ; Reussner, Ralf: Architecture-Based Change Impact Analysis in Information Systems and Business Processes. In: *Proceedings of the IEEE International Conference on Software Architecture*, 2017, S. 179–188

[RRG11] Raschke, Christian ; Roemer, Stephan ; Grossekatthoefer, Karsten: Test bed for Verification of Attitude Control System / Astro- und Feinwerktechnik Adlershof GmbH. Version: 2011. `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7990416`. 2011. – Forschungsbericht

[RS03] Robertson, Brent ; Stoneking, Eric: Satellite GN & C anomaly trends. In: *Advances in the Astronautical Sciences* 113 (2003), S. 531–542

[RSHR15] Rostami, Kiana ; Stammel, Johannes ; Heinrich, Robert ; Reussner, Ralf: Architecture-based assessment and planning of change requests. In: *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, 2015, S. 21–30

[SBB+10] Schaefer, Ina ; Bettini, Lorenzo ; Bono, Viviana ; Damiani, Ferruccio ; Tanzarella, Nico: Delta-oriented programming of software product lines. In: *Proceedings of the 14th International Conference on Software Product Lines*, 2010, S. 77–91

[Sch95] Schott, Jason R.: *Fault tolerant design using single and multicriteria genetic algorithm optimization*, Massachusetts Institute of Technology, Diss., 1995. `http://hdl.handle.net/1721.1/11582`

[SF05] Stigler, Hubert ; Felbinger, Günter: Der Interviewleitfaden im qualitativen Interview. In: Stigler, Hubert (Hrsg.) ; Reicher, Hannelore (Hrsg.): *Praxisbuch Empirische Sozialforschung in den Erziehungs- und Bildungswissenschaften*. Innsbruck : StudienVerlag, 2005, S. 128–135

[SFFW15] Savarino, Shirley ; Fitz, Rhonda ; Fesq, Lorraine ; Whitman, Gerek: Fault Management Architectures and the Challenges of Providing Software Assurance. In: *Proceedings of the 31st Space Symposium, Technical Track*, 2015

[SK09] Srivastava, Dr. Praveen R. ; Kim, Tai-hoon: Application of genetic algorithm in software testing. In: *International Journal of software Engineering and its Applications* 3 (2009), November, Nr. 4, S. 87–96

[SLW+14] Spörl, Andreas K. ; Lenzen, Christoph ; Wörle, Maria T. ; Hartung, Jens H. ; Mrowka, Falk ; Braun, Armin ; Wickler, Martin: Mission Planning System for the TET-1 On-Orbit Verification Mission. In: *Proceedings of the 2014 SpaceOps Conference*, 2014, S. 1–11

[SRC+12] Schaefer, Ina ; Rabiser, Rick ; Clarke, Dave ; Bettini, Lorenzo ; Benavides, David ; Botterweck, Goetz ; Pathak, Animesh ; Trujillo, Salvador ; Villela, Karina: Software diversity: state of the art and perspectives. In: *Software Tools for Technology Transfer* 14 (2012), Juli, Nr. 5, S. 477–495. http://dx.doi.org/10.1007/s10009-012-0253-y. – DOI 10.1007/s10009–012–0253–y

[SS13] Shrivastava, S. ; Sharma, A.: An approach for QoS based fault reconfiguration in service oriented architecture. In: *Proceedings of the 2013 International Conference on Information Systems and Computer Networks*, 2013, S. 180–184

[TKB+14] Thüm, Thomas ; Kästner, Christian ; Benduhn, Fabian ; Meinicke, Jens ; Saake, Gunter ; Leich, Thomas: FeatureIDE: An extensible framework for feature-oriented software development. In: *Science of Computer Programming* 79 (2014), S. 70–85. http://dx.doi.org/10.1016/j.scico.2012.06.002. – DOI 10.1016/j.scico.2012.06.002

[VG07] Voelter, Markus ; Groher, Iris: Product line implementation using aspect-oriented and model-driven software development. In: *Proceedings of the 11th International Software Product Line Conference*, 2007, S. 233–242

[WHWSH15] Winner, Hermann ; Hakuli, Stephan ; Wolf, Gabriele ; Singer (Hrsg.), Christina: *Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 3. Springer Vieweg, 2015. http://dx.doi.org/10.1007/978-3-658-05734-3. http://dx.doi.org/10.1007/978-3-658-05734-3. – ISBN 978–3–658–05733–6

[WPCE17] Walsh, Anthony ; Pecchioli, Mauro ; Carranza, Juan M. ; Ellsiepen, Peter: The Use of Model-Based Engineering Methodologies in Complex Ground Data Systems. In: *Space Operations: Contributions from the Global Community*. Springer-Verlag, 2017, S. 409–423

[YTRM10] Yoon, Zizung ; Terzibaschian, Thomas ; Raschke, Christian ; Maibaum, Olaf: Robust and Fault Tolerant AOCS of the TET Satellite. In: Sandau, Rainer (Hrsg.) ; Roeser, Hans-Peter (Hrsg.) ; Valenzuela, Arnoldo (Hrsg.): *Small Satellite Missions for Earth Observation.* Berlin, Heidelberg : Springer, 2010. – ISBN 978–3–642–03501–2, S. 401–410

[ZHC+14] Zolghadri, Ali ; Henry, David ; Cieslak, Jérôme ; Efimov, Denis ; Goupil, Philippe: *Fault diagnosis and fault-tolerant control and guidance for aerospace vehicles.* Springer-Verlag, 2014. – ISBN 978–1–4471–5312–2

[ZSG79] Zelkowitz, Marvin V. ; Shaw, Alan C. ; Gannon, John D.: *Principles of Software Engineering and Design.* Prentice Hall Professional Technical Reference, 1979. – ISBN 013710202X

The Karlsruhe Series on Software Design and Quality

Edited by Prof. Dr. Ralf Reussner // ISSN 1867-0067

Band 1 Steffen Becker
Coupled Model Transformations for QoS Enabled Component-Based Software Design.
ISBN 978-3-86644-271-9

Band 2 Heiko Koziolek
Parameter Dependencies for Reusable Performance Specifications of Software Components.
ISBN 978-3-86644-272-6

Band 3 Jens Happe
Predicting Software Performance in Symmetric Multi-core and Multiprocessor Environments.
ISBN 978-3-86644-381-5

Band 4 Klaus Krogmann
Reconstruction of Software Component Architectures and Behaviour Models using Static and Dynamic Analysis.
ISBN 978-3-86644-804-9

Band 5 Michael Kuperberg
Quantifying and Predicting the Influence of Execution Platform on Software Component Performance.
ISBN 978-3-86644-741-7

Band 6 Thomas Goldschmidt
View-Based Textual Modelling.
ISBN 978-3-86644-642-7

Band 7 Anne Koziolek
Automated Improvement of Software Architecture Models for Performance and Other Quality Attributes.
ISBN 978-3-86644-973-2

Band 8 Lucia Happe
Configurable Software Performance Completions through Higher-Order Model Transformations.
ISBN 978-3-86644-990-9

Band 9 Franz Brosch
Integrated Software Architecture-Based Reliability Prediction for IT Systems.
ISBN 978-3-86644-859-9

Band 10 Christoph Rathfelder
Modelling Event-Based Interactions in Component-Based Architectures for Quantitative System Evaluation.
ISBN 978-3-86644-969-5

Band 11 Henning Groenda
Certifying Software Component Performance Specifications.
ISBN 978-3-7315-0080-3

Band 12 Dennis Westermann
Deriving Goal-oriented Performance Models by Systematic Experimentation.
ISBN 978-3-7315-0165-7

Band 13 Michael Hauck
Automated Experiments for Deriving Performance-relevant Properties of Software Execution Environments.
ISBN 978-3-7315-0138-1

Band 14 Zoya Durdik
Architectural Design Decision Documentation through Reuse of Design Patterns.
ISBN 978-3-7315-0292-0

Band 15 Erik Burger
Flexible Views for View-based Model-driven Development.
ISBN 978-3-7315-0276-0

Band 16 **Benjamin Klatt**
Consolidation of Customized Product Copies into Software Product Lines.
ISBN 978-3-7315-0368-2

Band 17 **Andreas Rentschler**
Model Transformation Languages with Modular Information Hiding.
ISBN 978-3-7315-0346-0

Band 18 **Omar-Qais Noorshams**
Modeling and Prediction of I/O Performance in Virtualized Environments.
ISBN 978-3-7315-0359-0

Band 19 **Johannes Josef Stammel**
Architekturbasierte Bewertung und Planung von Änderungsanfragen.
ISBN 978-3-7315-0524-2

Band 20 **Alexander Wert**
Performance Problem Diagnostics by Systematic Experimentation.
ISBN 978-3-7315-0677-5

Band 21 **Christoph Heger**
An Approach for Guiding Developers to Performance and Scalability Solutions.
ISBN 978-3-7315-0698-0

Band 22 **Fouad ben Nasr Omri**
Weighted Statistical Testing based on Active Learning and Formal Verification Techniques for Software Reliability Assessment.
ISBN 978-3-7315-0472-6

Band 23 **Michael Langhammer**
Automated Coevolution of Source Code and Software Architecture Models.
ISBN 978-3-7315-0783-3

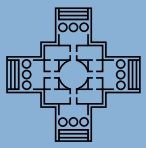
Band 24 Max Emanuel Kramer
Specification Languages for Preserving Consistency between Models of Different Languages.
ISBN 978-3-7315-0784-0

Band 25 Sebastian Michael Lehrig
Efficiently Conducting Quality-of-Service Analyses by Templating Architectural Knowledge.
ISBN 978-3-7315-0756-7

Band 26 Georg Hinkel
Implicit Incremental Model Analyses and Transformations.
ISBN 978-3-7315-0763-5

Band 27 Christian Stier
Adaptation-Aware Architecture Modeling and Analysis of Energy Efficiency for Software Systems.
ISBN 978-3-7315-0851-9

Band 28 Lukas Märtin
Entwurfsoptimierung von selbst-adaptiven Wartungsmechanismen für software-intensive technische Systeme.
ISBN 978-3-7315-0852-6



The Karlsruhe Series on Software Design and Quality

Edited by Prof. Dr. Ralf Reussner

In heutigen technischen Systemen werden Hardwareressourcen in homogener redundanter Ausprägung vorgesehen. Trotz etablierter Fehlerbehandlungsstrategien unterliegt die selbst-adaptive Wartung natürlichen Limitierungen, die die Beteiligung eines Entscheidungsträgers erfordern. Dies trifft insbesondere dann zu, wenn neben der Funktionalität eine optimierte Systemgüte als wesentlich für einen langfristigen Betrieb gilt.

Diese Arbeit stellt neuartige Konzepte zur effizienten Entscheidungsunterstützung in der Rekonfiguration software-intensiver technischer Systeme mit limitiertem Wartungszugriff vor. Entgegen rein redundanzorientierter Ansätze, basiert die verfolgte Methodik auf der prädiktiven Vorausberechnung adäquater Konfigurationsalternativen im relevanten Lösungsraum. Das Wissen über Konfigurationsbeziehung wird frühzeitig in einem Expertensystem manifestiert, welches zur Laufzeit kosteneffiziente Alternativen autark erhebt und die Anpassungsmöglichkeiten transparent darstellt.

ISSN 1867-0067
ISBN 978-3-7315-0852-6

Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-0852-6
9 783731 508526 >