Aufbau und Einsatz von Entwurfssprachen zur Auslegung von Satelliten

Von der Fakultät für Luft- und Raumfahrttechnik und Geodäsie
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung.

Vorgelegt von
Dipl.-Ing. Johannes Groß
geboren in Reubach

Hauptberichter:	PD. Dr.-Ing. Stephan Rudolph
Mitberichter:	Prof. Dr.-rer.-nat. Hans-Peter Röser
Tag der mündlichen Prüfung:	22.10.2013

Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen
Universität Stuttgart
2014

Vorwort

Diese Arbeit entstand während meiner Zeit in der Arbeitsgruppe Ähnlichkeitsmechanik am Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen. Da ich nach dem Studium in der Vertiefungsrichtung Datenverabeitung auch meine Studien- und Diplomarbeit im Umfeld der sogenannten Pi-Gruppe absolviert habe, geht eine lange Zeit der Zusammenarbeit, des gemeinsamen Lernens und Forschens zu Ende. In dieser Zeit gab es viele Menschen, die mich unterstützt haben und mit denen ich zusammen wachsen, diskutieren und lernen konnte.

Zuerst danke ich meinem Doktorvater und Hauptberichter Herrn PD. Dr.-Ing. Stephan Rudolph. Er hat mit den Entwurfssprachen eine neuartige Methodik für den Ingenieurentwurf und ein grossartiges Forschungsgebiet für seine Arbeitsgruppe geschaffen. Ich danke vor allem für die engagierte Betreuung, die er mir in der Zeit der Dissertation geboten hat.

Ich danke Herrn Prof. Dr. rer. nat. Hans-Peter Röser für die Übernahme des Mitberichts und die zahlreichen Kooperationen mit seinem Institut.

Herrn Prof. Dr.-Ing. habil. Bernhard Weigand, der Deutschen Forschungsgemeinschaft und dem Graduiertenkolleg 1095 danke ich für ein Stipendium über drei Jahre und interessante gemeinsame Tagungen.

Meinen Kollegen und den Studenten in der Arbeitsgruppe Ähnlichkeitsmechanik danke ich für eine gute gemeinsame Zeit und die vielen Beiträge zu dieser Arbeit. Dabei geht mein Dank insbesondere an Peter Arnold für viele gemeinsame Entwicklungen und seine Editoren, Johannes Beichter für viele gute Zeilen Code, Johannes Bürkle für die Mitarbeit am Routing, Michael Bölling für die gemeinsame Arbeit an den Heatmaps, Marc Eheim für das Packaging und Routing, Stefan Hess für die Pi-Analysen und den Support, Martin Motzer für die Visualisierungen, Axel Reichwein für seine Vorarbeiten zum Einsatz der Modellierungssprachen und Samuel Vogel für gute Diskussionen. Christian Messe danke ich für die Zusammenarbeit zur Thermalanalyse und eine spannende Zeit im GRK. Ich danke den Studien- und Diplomarbeitern Felix Jakob, Muhammed Sahin Kocak, Marius Riestenpatt genannt Richter, Jens Schmidt und Roland Weil für sehr gute Arbeiten und viel eingebrachtes Engagement. Jeder Einzelne hat zum Gelingen dieser Arbeit beigetragen.

Meiner Familie und meinen Freunden danke ich von ganzem Herzen für Ihre Unterstützung.

Stuttgart, Oktober 2013 Johannes Groß

Inhaltsverzeichnis

Abkürzungen

AOCS	Attitude and Orbit Control System (hier: Lageregelungssystem)
B-Rep	Boundary Representation
CAD	Computer Aided Design
CDF	Concurrent Design Facility
CE	Concurrent Engineering
CEF	Concurrent Engineering Facility
CSG	Constructive Solid Geometry
ESA	European Space Agency
INCOSE	International Council on Systems Engineering
LHS	Left-Hand-Side
LNA	Low-Noise-Amplifier
MLI	Multi-Layer-Insulation
OBDH	On-Board Data Handling
OCC	OpenCascade
OCL	Object Constraint Language
OMG	Object Management Group
RHS	Right-Hand-Side
SysML	Systems Modeling Language
TWT	Travelling Wave Tube Amplifier (Wanderfeldröhre)
TTC	Telemetry, Tracking and Command (hier: Kommunikationssystem)
UML	Unified Modeling Language
XML	Extended Markup Language

Begriffe

Systemtopologie:	Kombination aus Komponenten, die ein funktionierendes System darstellen.
Entwurfsraum:	Raum, in dem alle topologischen (i.S.d. Systemtopologie) und parametrischen Variationsmöglichkeiten eines Entwurfs enthalten sind.

Konventionen

UML-Klassennamen werden in *Italic* und der Anfangsbuchstabe wird groß geschrieben.
UML-Attribute werden in `schreibmaschinenschrift` und klein geschrieben.

Kurzfassung

Die vorliegende Arbeit zeigt den Aufbau graphenbasierter Entwurfssprachen auf Basis der Unified Modeling Language (UML) im Einsatz für die Auslegung von Satelliten. Bei der Auslegung moderner Satelliten wird in einem von den Anforderungen implizit festgelegten Entwurfsraum ein Optimum gesucht. Um in Zukunft bei gleich bleibender Entwurfsqualität mehr Entwürfe in kürzerer Zeit zu geringeren Kosten untersuchen zu können, wird in dieser Arbeit der automatische Entwurf von Satelliten am Beispiel des FireSat [69] vorgestellt.

Für den automatischen Entwurf wird das Entwurfswissen über die Auslegung und die Integration der einzelnen Systemkomponenten in einer Hierarchie verschiedener Entwurfssprachen modular abgebildet. Die Kopplungen zwischen den Systemkomponenten werden für die Auslegung aufgelöst und als generische Schnittstellen formuliert. Die Integration der einzelnen Komponenten zu einem Gesamtsystem erfolgt nach vordefinierten Regeln. Die gesamte Abfolge der Entwurfsregeln, vom funktionalen Vorentwurf bis hin zur Gesamtintegration mit den funktionalen Nachweisen durch die entsprechenden numerischen Simulationsmodelle, stellt eine über verschiedene, variierende Anforderungen robuste Entwurfssequenz dar.

In der Vorentwurfsphase werden die Gleichungen bottom-up von Komponenten-Ebene über die Subsystem-Ebene zum Gesamtsystem regelbasiert zusammengesetzt und gelöst. Für die genauere Untersuchung bestimmter Auslegungspunkte wird automatisch eine Sensitivitätsanalyse über die wesentlichen Entwurfsgrößen des Gesamtentwurfs erstellt. Für den Vergleich verschiedener Satellitentopologien werden in dieser Arbeit über 3300 topologisch (d.h. welche der Komponenten) wie auch parametrisch (d.h. wie speziell dimensioniert) unterschiedliche Satelliten vollautomatisch ausgelegt. Daraus werden Pareto-Flächen gebildet anhand derer der topologische und der parametrische Entwurfsraum analysiert wird. Der im Laufe der Regelausführung entstehende Entwurfsgraph repräsentiert ein durchgängiges, ganzheitliches und konsistentes Datenmodell, das die Integration von funktionaler, geometrischer und physikalischer Modellierung im Format der Unified Modeling Language (UML) vereinigt.

Die Umsetzung der vordefinierten Entwurfshandlungen (z.B. von Konstruktionsvorgängen wie dem Generieren von Halterungslaschen oder Taschen) für den FireSat erfolgt unter Berücksichtigung aller gültigen Kopplungen zwischen den erzeugten multi-disziplinären Entwurfsinformationen. Die Multi-Disziplinarität des FireSat-Beispiels deckt die funktionale Auslegung, die Generierung von Geometrie in CATIA V5 und OpenCascade, die Thermalsimulation mit ESATAN-TMS und die regelungstechnische Lage-Simulation mit Simulink ab. Das Auslegungsbeispiel FireSat wird durch eine Machbarkeitsstudie zur automatisierten Anordnung (Packaging) und durch die Integration der automatischen Verkabelung (Routing) im Satellit ergänzt.

Mit diesem Einsatz der Entwurfssprachen ist die Machbarkeit eines automatisierten Satellitenentwurfs durch Existenz bewiesen. Im Rahmen der Arbeit ist dabei keine prinzipielle theoretische Grenze für einen vollständig automatischen Entwurf sichtbar geworden. Durch die rigorose formale Abbildung und die daraus folgende algorithmische Verarbeitung des Entwurfswissens, ist die für einen Entwurf erforderliche Zeit auf die Programmlaufzeiten der Algorithmen reduziert. Damit ist hinsichtlich der gewählten Modellvorstellungen und Algorithmen die theoretische Untergrenze für die Dauer des Entwurfsprozesses erreicht.

Abstract

The present thesis describes the conception of graph-based design languages using the Unified Modeling Language (UML) for satellite design. Nowadays designing a satellite can be described as a search for an optimum in a design space implicitly defined by requirements. Supporting the analysis of more designs in shorter time with the same quality this work illustrates the automated design of satellites using the example of the FireSat mission [70].

Automating the design process requires the domain knowledge about the layout and the integration of the systems´ components. The domain knowledge is mapped into a hierarchy of different modular design languages. For the layout the couplings between the systems´ components are resolved and defined as generic interfaces. A sequence of predefined rules specifies the integration of the components to a satellite system. Ranging from preliminary design to the system integration, the sequence of rules also includes the verification by means of detailed numerical simulation models. The resulting automated design process is robust under several varying initial requirements.

In the preliminary design phase the analytical descriptions of the satellites´ components are assembled automatically bottom-up from component-level to subsystem-level to system-level. Solving the resulting system of equations yields the satellites´ design parameters. Analysing the sensitivity of all relevant design parameters automatically allows for the investigation of a certain design point. Laying out more than 3300 topological (i.e. which components) and parametric (i.e. how dimensioned) different satellites, enables the comparison of distinct system topologies. Building Pareto-surfaces from these design points illustrates the topological and parametric FireSat design space. Executing the rule sequence creates a design graph representing an integrated and consistent data model along the entire design process. This design graph allows for the integration of geometric, functional and physical modeling in a consistent central model based on the unified modeling language (UML).

Executing predefined rules to create mounting links to satellite electronic boxes or to create stiffening corrugations in the satellite structure, is performed with respect to the relevant couplings given by the multi-disciplinary design information. Being a real multi-disciplinary example, the FireSat design process ranges from functional layout to geometry generation in CATIA V5 and OpenCascade to thermal simulation in ESATAN-TMS to an attitude simulation in Simulink. Incorporating also a feasibility study for an automated packaging and using an automated routing of the satellites cables completes the portfolio of design automation.

The feasibility of an automated satellite design is proven by evidence in this work. Regarding the full automation of satellite design, in the scope of this work no principal theoretical limit became apparent. By the rigorous mapping of the design knowledge to a formal description, the consequent automated processing reduces the duration for a single satellite design to the execution times of the different algorithms. With regard to the chosen models and algorithms thus the theoretical lower limit for the duration of the design process is reached.

1 Herausforderung Satellitenentwurf

Moderne Satelliten erfüllen autark in der Umgebung des Weltraums eine Vielzahl an Funktionen. Die Funktionen werden an Bord des Satelliten von verschiedenen Subsystemen und Komponenten erfüllt. So gibt es zum Beispiel ein System für die Energieversorgung oder das Kommunikationssystem. Die Subsysteme können für jede Funktion auf unterschiedliche Lösungsprinzipien zurückgreifen, so kann die Energie zum Beispiel photovoltaisch oder aus Radioaktivität gewonnen werden. Die Lösungsprinzipien werden nach den Randbedingungen ausgelegt und als Komponenten umgesetzt. Durch die alternativen Lösungsprinzipien und deren Auslegung nach den Randbedingungen wird ein topologischer und parametrischer Entwurfsraum aufgespannt.

Jeder zulässige Punkt in diesem Entwurfsraum ist ein potentieller Kandidat für die Lösung der Entwurfsaufgabe. Die Punkte besitzen jedoch unterschiedliche Qualitäten bezüglich der Anforderungen an den Entwurf. Um die Qualitäten eines Punktes im Entwurfsraum vollständig zu erfassen, muss der dazugehörige Satellit vollständig ausgelegt werden. Durch die Anforderung, einen guten Entwurf schnell und zu möglichst geringen Kosten in diesem Entwurfsraum zu finden, entsteht die besondere Herausforderung im heutigen Satellitenentwurf.

1.1 Entwurfsprozess

Aus den Erfahrungen vieler verschiedener Raumfahrtprojekte wurde von der Europäischen Raumfahrtagentur ESA eine Standardisierung für die verschiedenen Projektphasen entwickelt. Die Standardisierung wurde eingeführt um „technische Risiken, Zeitplanungsrisiken und ökonomische Risiken in Projekten zu minimieren“ [14]. In Abb. 1.1 sind die Projektphasen eines Raumfahrtprojektes nach [14] abgebildet.

Die Missionsanalyse, die Funktionsbeschreibung (Phase 0+A) und das Festlegen der Anforderungen (Phase B) werden hier als Vorentwurf bezeichnet. Innerhalb des Vorentwurfs hat sich über die vergangenen Jahre das sogenannte Concurrent Engineering als Methode etabliert. Beim Concurrent Engineering wird der Entwurf der einzelnen Subsysteme eines Satelliten parallel durchgeführt. Die verschiedenen Experten arbeiten zeitgleich an der Lösung und kommunizieren die Auswirkungen auf die anderen Subsysteme untereinander.

Dafür wird durch sogenannte Concurrent Design Facilities (CDF/CEF) (u.a. [1, 7, 54, 66]) eine Infrastruktur zur Verfügung gestellt, in der das Team arbeiten kann. Ein Concurrent Design Facilitiy besteht aus einem Raum mit vernetzten Computern und verschiedenen Präsentations- und Simulationstools sowie meist Excel oder einer Datenbank zum Datenaustausch.

In der Entwurfsphase des Detailentwurfs (C) wird die Konstruktionsarbeit ausgeführt und ein detailliertes Geometriemodell erstellt. Auf Basis des Geometriemodells sowie der funktionalen

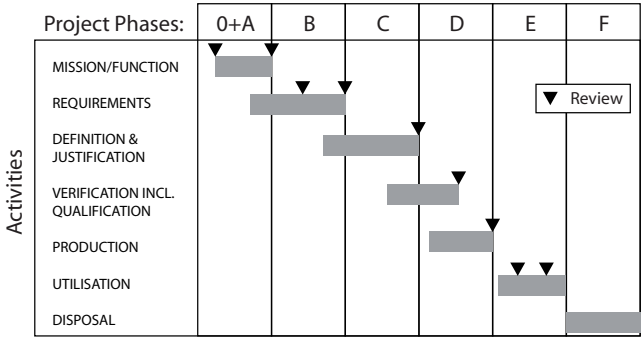


Abbildung 1.1: Projektphasen in der Raumfahrt nach [14]

und der operationellen Beschreibung des Entwurfs werden zur Verifikation des Entwurfs in den verschiedenen Ingenieurdomänen umfangreiche Simulationen durchgeführt. Die Simulationen umfassen unter anderem Feldprobleme, wie eine Thermalsimulation oder eine Finite-Elemente Simulation außerdem funktionale Aspekte, wie die Zuverlässigkeit und auch dynamische Aspekte der Steuerung und Regelung. Nach der erfolgreichen Verifikation und Validierung kann der Satellit gebaut (Phase D) und betrieben werden (Phase E). Mit der Entsorgung in Phase (F) ist der gesamte Lebenszyklus des Satelliten abgeschlossen.

1.2 Problematik

Bei der Beurteilung eines Punktes im Entwurfsraum kann nur durch eine umfassende Auslegung eines Entwurfs sichergestellt werden, dass nicht in einer späten Entwurfsphase verdeckte Kosten auftreten. Für die umfassende Auslegung müssen die Phasen 0-C durchschritten werden.

Im Vorentwurf wird üblicherweise eine Systemtopologie für den Satelliten aus der Vielfalt der kombinatorischen Möglichkeiten ausgewählt. Eine fundierte Entscheidung kann nur auf der Basis physikalischer Analysen gefällt werden. Dafür müssen die Auslegungsgleichungen der verschiedenen Lösungsprinzipien zu einem Gesamtsystem aggregiert und gelöst werden. Die Exploration des entstehenden Entwurfsraums wird manuell durchgeführt, Vollständigkeit ist daher nur sehr schwer zu erreichen.

Die einzelnen Arbeitsschritte auf dem Weg zu einer detaillierten Entwurfsbeschreibung werden für die gegebenen Randbedingungen bisher zumeist manuell ausgeführt. Bei Änderungen in vorhergehenden Entwurfsschritten werden diese Arbeitsschritte zum Teil oder gänzlich neu ausgeführt. Anhand verschiedener Simulationsmodelle werden die vielfältigen Kopplungen zwischen den einzelnen Komponenten des Systems analysiert. Der Aufbau dieser Simulationsmodelle ist ein aufwendiger Vorgang, da in jedes Modell viele Randbedingungen aus dem aktuellen Entwurfsstand eingehen. Bei Änderungen am aktuellen Entwurfsstand müssen alle Modelle angepasst und zueinander konsistent gehalten werden.

Die Entwurfsbeschreibung ist entlang des Entwurfsprozesses über verschiedene Dokumente und Modelle verteilt, die manuell untereinander abgeglichen werden müssen. Der manuelle Abgleich ist aufwendig und birgt ein zusätzliches Fehlerrisiko für den Entwurf. Werden entlang des Entwurfsprozesses Probleme im Entwurf erkannt, so ist es aufgrund der aufwendigen Modellierung wichtig, eine entlang des Entwurfsprozesses „lokale“ Lösung des Problems zu finden, die möglichst wenig Auswirkungen auf andere Systeme, Modelle und vorgelagerte Entwurfsschritte hat. Aufgrund endlicher Ressourcen (Zeit, Geld, etc.) können im Rahmen des herkömmlichen Entwurfsprozesses nur wenige Varianten untersucht werden.

Die Problematik lässt sich zusammenfassend auf eine fehlende einheitliche Methodik für die Abbildung und Automatisierung des gesamten Entwurfsprozesses zurückführen. Um eine adäquate Lösung für diese Problematik zu finden, werden in dieser Arbeit die bisherigen Unterteilungen des manuellen Entwurfsprozesses (vgl. Abb. 1.1) außer Acht gelassen. Es wird eine Methode benötigt, die über den gesamten Entwurfsprozess hinweg verwendet werden und die alle darin vorkommenden Modellvorstellungen abbilden kann. Die Methode der graphenbasierten Entwurfssprachen nach [36] verspricht eine solche einheitliche Beschreibung des gesamten Entwurfsprozesses.

1.3 Aufgabenstellung

In dieser Arbeit wird für die Demonstration einer einheitlichen, durchgängigen Methode der Aufbau und Einsatz von Entwurfssprachen für die Auslegung von Satelliten vorgestellt. Dafür werden in einer Beispielauslegung der in [70] dokumentierten FireSat-Mission die oben genannten Probleme genauer beleuchtet und mittels der Entwurfssprachen Lösungen entwickelt.

Für den Vorentwurf werden die Gleichungen für die Beschreibung der Komponenten und Systemzusammenhänge automatisch zusammengesetzt. Anhand des entstehenden Gleichungssystems kann eine Systemtopologie untersucht werden. Durch die Variation der Randbedingungen wird die automatische Exploration von Entwurfsräumen auf Komponenten-, Subsystem- und Systemebene durchgeführt. Für den Aufbau der detaillierten Entwurfsbeschreibung einschließlich des Geometriemodells werden die bisher manuellen Auslegungsschritte automatisiert durchgeführt. Die einzelnen Subsysteme sowie die automatisierten Auslegungsschritte sind dabei an sich unabhängig voneinander wiederverwendbar. Die Simulationsmodelle für den ausgearbeiteten Entwurf werden aus der Entwurfssprache heraus generiert. Dabei wird ein CAD-Modell, ein Thermalmodell und eine Simulation des Systemverhaltens enthalten sein, um die unterschiedlichen Modellbereiche von Geometrie, Physik (Feldprobleme) und auch Regelungsaspekten abzudecken.

Diese drei Bereiche werden mit den Entwurfssprachen in einer durchgängigen Methode umgesetzt. Für den Entwurf des FireSat wird ausgehend von den Anforderungen über die Vorauslegung bis zu den detaillierten Beschreibungen eine automatische Prozesskette entwickelt. Durch die vollständige Automatisierung des Entwurfsprozesses reduziert sich die Dauer für die Auslegung einer Variante des FireSat auf die Programmlaufzeiten der Algorithmen und stellt damit die theoretische Untergrenze für die Entwurfszeit dar.

2 Grundlagen

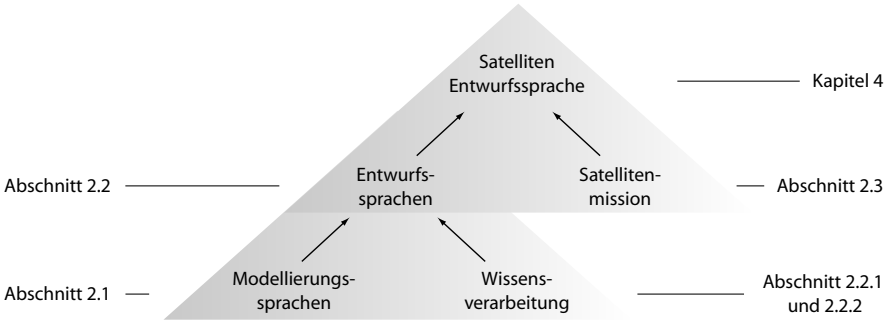


Abbildung 2.1: Aufbau der Grundlagen

Die vorliegende Arbeit baut auf Grundlagen aus mehreren Wissensgebieten auf und führt diese in einer graphenbasierten Entwurfssprache erfolgreich zusammen. In Abb. 2.1 sind diese Gebiete, und die Abschnitte in denen sie erklärt werden, dargestellt. Die Modellierungssprachen aus der Softwaretechnik werden verwendet um die Beschreibung des Satelliten zu formalisieren. Die Methodik der Entwurfssprachen baut mit der Verwendung der Modellierungssprachen auf diesen Konzepten auf. Des weiteren beinhalten die Entwurfssprachen Konzepte zur automatischen Wissensverarbeitung. Der Kern der Arbeit, die Satelliten-Entwurfssprache, entsteht aus der Anwendung der Methodik der Entwurfssprachen auf die Auslegung einer Satellitenmission.

2.1 Modellierung mit UML

Die Modellierungssprachen entstammen dem Bereich des Software Engineering. Eine Modellierungssprache basiert auf einer formalen Spezifikation der Sprachelemente. Diese Spezifikation wird auch Metamodell genannt. Auf der Basis eines Metamodells ist es möglich, eine einheitliche Verarbeitung der Sprachelemente über verschiedene Tools hinweg zu garantieren. Es gibt verschiedene weit verbreitete Modellierungssprachen, darunter z.B. die Unified Modeling Language (UML) [47], die Systems Modeling Language (SysML) [46] und Ecore. In dieser Arbeit wird die UML verwendet.

UML. Die UML wird von der Object Management Group (OMG) spezifiziert. In dieser Arbeit wird nur ein sehr kleiner Teil des Sprachumfangs der UML zur Modellierung verwendet. Dies sind im Wesentlichen auf Klassenebene Klassen, Attribute von Klassen und verschiedene Verbindungen zwischen Klassen (Assoziationen und Vererbungen). Auf Instanzebene sind es die Instanzen, Slots für die Eingabe von konkreten Werten und Links für die Verbindungen. Die Elemente werden hier kurz vorgestellt:

Klassen. Klassen dienen als allgemeine Schablonen für eine Gruppe von Objekten. Eine Klasse beschreibt nach der UML-Spezifikation [47] eine Menge an Objekten mit denselben Eigenschaften, Randbedingungen und Bedeutungen[1]. Eigenschaften (features) und Randbedingungen (constraints) sind spezifizierte UML-Elemente, die Bedeutung (semantics) einer Klasse wird in der UML nicht formal abgebildet.

Eine Klasse kann mit anderen Klassen über Assoziationen und Vererbungen verbunden werden. Eine Assoziation drückt eine „hat ein“-Relation aus und eine Vererbung eine „ist ein“-Relation. Die Klassen in Abb. 2.2 lassen also die Aussage zu: „Ein Kommunikationssystem hat eine Antenne und eine Hornantenne ist eine Antenne“. Da auch eine *Parabolantenne* eine *Antenne* ist, kann ein *Kommunikationssystem* nach Abb. 2.2 entweder mit einer *Parabol-* oder mit einer *Hornantenne* ausgeführt werden.

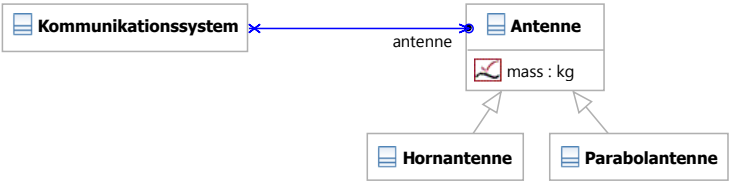


Abbildung 2.2: Beispiel von Klassen

Beim Einsatz in Entwurfssprachen werden innerhalb der Klassen Gleichungen mittels Constraints spezifiziert. Diese Gleichungen verknüpfen die Attribute der Klassen (wie z.B. die `mass` der *Antenne*) miteinander und gelten für jede Instanz einer Klasse.

Instanzen. Für die konkrete Ausprägung einer Klasse wird eine Instanz erzeugt. Eine Instanz ist über eine „ist vom Typ“-Relation mit einer Klasse verbunden. In Abb. 2.3 sind zwei Instanzen abgebildet. Die obere Instanz „hkSystem“ ist vom Typ *Kommunikationssystem*. Sie ist mit der unteren Instanz „a012“ vom Typ *Hornantenne* über einen Link verbunden. Durch die Vererbung von der *Antenne* kann die *Hornantenne* mit dem *Kommunikationssystem* verbunden werden.

[1] „A class describes a set of objects that share the same specifications of features, constraints, and semantics“ [47].

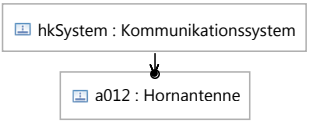


Abbildung 2.3: Beispiel zweier Instanzen

2.2 Entwurfssprachen

Die Idee der Entwurfssprachen für den Einsatz im allgemeinen Ingenieurentwurf geht zurück auf Rudolph [55]. Dort entstand der Gedanke aus der Kombination zweier Vorarbeiten. Zum einen ist dies der Aufbau komplexer Strukturen aus einfachen Regeln nach Lindenmayer [41,45]. Zum anderen ist es ein Algorithmus zur automatischen Lösung des Zuordnungsproblems bei Gleichungssystemen (welche Variable wird durch welche Gleichung gelöst) [72, 73]. Diese beiden Ansätze wurden von Rudolph und Noser erstmals in [60] und [45] kombiniert und bilden damit den Grundstein zum regelbasierten Aufbau großer Gleichungssysteme.

Die Methodik der Entwurfssprachen wurde von Alber und Rudolph am Beispiel von Strommasten [2, 58] und Raumstationen [3] weiterentwickelt. In [2] kommt zum ersten Mal die Generierung von Modellen in Anwendungsprogrammen hinzu.

Die erste Implementierung eines Entwurfscompilers [26], der die Entwurfssprachen verarbeitet, erfolgte in C++ auf der Basis eines XML Datenmodells. Mit diesem Entwurfscompiler wurden unter anderem Entwurfssprachen für den Entwurf von Raumstationen [27, 28], Satelliten [63, 64], Flugzeugrümpfen [34], Luftschiffen [59] und Fahrzeugkarosserien [25, 33] aufgebaut.

Einen wesentlichen Entwicklungsschritt in der Methodik der Entwurfssprachen stellt der Wechsel der Repräsentationsform von einem Entwurfssprachen-spezifischen XML-Format auf eine Repräsentation in der UML (mit Erweiterungen für die Abbildung der Entwurfssprachen) dar. Die Grundlage dafür schaffte Reichwein [52] mit der Datenintegration verschiedener Domänen und den Schnittstellen zu den entsprechenden Anwendungsprogrammen. Dies ermöglichte die Nutzung der objektorientierten Modellierungskonzepte für die Beschreibung des Entwurfs in einer Entwurfssprache. Neben den Arbeiten von Arnold [5], Beilstein [8], Landes [38, 39] und Vogel [68, 69] ist diese Arbeit eine der ersten unter der Verwendung der UML.

2.2.1 Informationsarchitektur

Die Informationsarchitektur einer Entwurfssprache wie in Abb. 2.4 dargestellt, weist einen wesentlichen Unterschied zum Informationsfluss im klassischen Entwurfsprozess auf. Während der Informationsfluss im klassischen Entwurfsprozess in Abb. 2.4 ganz rechts über die Geometrie in die verschiedenen anderen Domänenmodelle vertikal verläuft, wird der Entwurf in einer Entwurfssprache zunächst abstrakt in Vokabeln und Regeln abgebildet und dann von einem Entwurfscompiler in die verschiedenen Ansichten übersetzt.

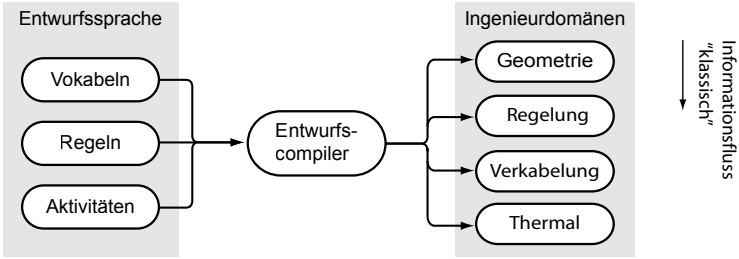


Abbildung 2.4: Informationsarchitektur der Entwurfssprachen

Diese Informationsarchitektur löst das CAD-Paradigma des klassischen Entwurfsprozesses auf und stellt die objekt-orientierte Modellierung mit Entwurfssprachen in den Vordergrund. Durch die Entwurfsrepräsentation in Vokabeln und Regeln wird dadurch eine Formalisierung des gesamten Entwurfsprozesses möglich.

Lösungspfad. In den Vokablen der Entwurfssprache können algebraische Gleichungen für die Beschreibung der Kopplungen verwendet werden. Für die Behandlung von Gleichungen in einer Entwurfssprache ist im Entwurfscompiler ein sogenannter Lösungspfadgenerator [9, 73] integriert. Dieser wertet das Gleichungssystem des gesamten Entwurfs aus und erstellt, basierend auf einem Graphenalgorithmus, einen Lösungspfad. Der Lösungspfad gibt an, in welcher Reihenfolge die einzelnen Gleichungen gelöst werden müssen. Für ein Gleichungssystem

$$a = b^2 + c^{-1}, \qquad d = a^2 + c^3, \qquad b = 4 \qquad \text{und} \qquad c = 5$$

ergibt sich der Lösungspfad zu der in Abb. 2.5 dargestellten Abfolge.

Abbildung 2.5: Lösungspfad des Beispiels

Die an diesem Beispiel noch einfach erscheinende Aufgabe, die Gleichungen in die richtige Anordnung zu bringen, kann durch die automatische Lösung des Zuordnungsproblems auch für den Entwurf eines komplexen Systems mit mehreren tausend Gleichungen automatisch durchgeführt werden. Dadurch kann die deklarative Wissensrepräsentation der Gleichungen über alle abgebildeten Hierarchiebenen hinweg durchgängig verarbeitet werden. Dies ermög-

licht zum Beispiel, dass bei einer Komponente, die nach den globalen Systemanforderungen ausgelegt wird, mit demselben Gleichungssystem auch die Auswirkungen von Änderungen an dieser Komponente auf die Systemanforderungen untersucht werden können.

2.2.2 Vokabeln und Regeln

Die Beschreibung des Entwurfs in einer Entwurfssprache erfolgt mittels Vokabeln und Regeln.

Vokabeln. Eine Vokabel in einer Entwurfssprache kann ein allgemeines Konzept im Entwurf eines komplexen Systems repräsentieren. In dieser Arbeit werden nur wenige Elemente der UML verwendet, um die Vokabeln des Satellitenentwurfs zu modellieren. Die am meisten verwendeten Elemente sind die Klassen, Instanzen, Attribute und Slots. Diese werden hauptsächlich über Assoziationen, Vererbungen und Links (auf Instanz-Ebene) miteinander verbunden.

Regeln. Die Regeln können in einer Entwurfssprache auf unterschiedliche Weise formuliert werden. Ganz allgemein lassen sich die Regeln in deklarative und imperative Formulierungen unterscheiden. Zu den deklarativ formulierten Regeln gehören die mathematischen Gleichungen und OCL-Constraints [48]. Als zentrales Element für den Aufbau eines Entwurfs in einer Entwurfssprache werden graphische Regeln verwendet. Auch diese zählen zu den deklarativ formulierten Regeln. Neben den graphischen Regeln kann auch Java-Code verwendet werden, um Modelltransformationen zu definieren. Die Java-Regeln stellen imperativ formulierte Regeln dar.

Die graphischen Regeln in einer Entwurfssprache beruhen auf einem modifizierten X-Schema nach [24], das von Alber [3] für die Verwendung in Entwurfssprachen angepasst wurde. Bei der Implementierung eines Entwurfscompilers auf Basis der Modellierungssprachen wurde das X-Schema (vier-Quadranten-Schema) zu einem Left-Hand-Side / Right-Hand-Side (zwei-Quadranten-Schema) vereinfacht. In Abb. 2.6 ist eine graphische Regel abgebildet, die

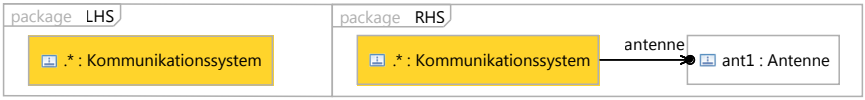


Abbildung 2.6: Einfache Regel

eine Antenne an das Kommunikationssystem anfügen soll. Das Grundprinzip bei der Suche auf der linken Seite der Regel ist, dass alles was angegeben ist, als Bedingung mit in die Suche eingeht. Alles was nicht angegeben ist, wird nicht beachtet. Die Suchseite sagt also aus: „Alle Instanzen, deren Namen dem regulären Ausdruck „.*“ entsprechen und deren Classifier *Kommunikationssystem* ist“. Man sucht also alle *Kommunikationssystem*-Instanzen. An all diese Instanzen wird durch die Regelausführung eine *Antenne* „ant1“ angehängt.

Aktivitäten. Die Abfolge der Regeln wird in einem Aktivitätsdiagramm festgelegt. Das Aktivtätsdiagramm wird mit UML-Elementen modelliert, die über Stereotypen um die Semantik der Entwurfssprachen erweitert sind. In Abb. 2.7 ist ein einfaches Aktivitätsdiagramm abgebildet.

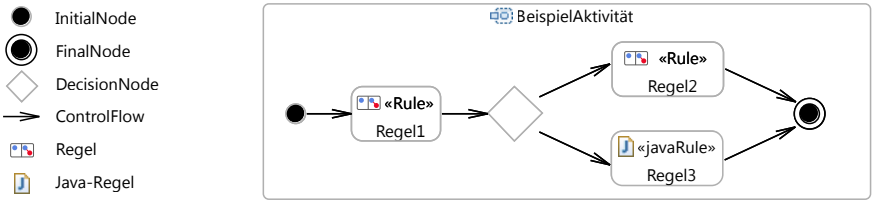


Abbildung 2.7: Beispiel einer Aktivität

Der Ablauf des Aktivitätsdiagramms startet am InitialNode auf der linken Seite. Dann wird dem ControlFlow folgend die erste Regel „Regel1“ ausgeführt. Danach kommt ein DecisionNode. In dem DecisionNode ist eine Bedingung für die Entscheidung gespeichert, welchem ControlFlow gefolgt werden soll. Danach wird entweder die obere Regel „Regel2“ oder die unter Java-Regel ausgeführt. Bei der Java-Regel handelt es sich um Java-Code, der auf dem aktuellen Modell des Entwurfs ausgeführt wird. Am FinalNode endet diese Aktivität.

2.3 Satellitenentwurf

Als Grundlage für die Anwendung der Entwurfssprachen auf den Satellitenentwurf wird im Folgenden die Mission erläutert. Neben der Missionsbeschreibung werden noch einige verwandte Ansätze und vorhergehende Arbeiten vorgestellt.

2.3.1 FireSat Mission

Bei der ausgewählten Beispielmission handelt es sich um den sogenannten „FireSat“. Dies ist eine in [70] erstmals beschriebene, inzwischen weiter entwickelte [12, 13, 19–23, 40], fiktive Mission zum Auffinden, Überwachen und Verfolgen von Waldbränden in den USA und Kanada. In Abb. 2.8 ist die Mission skizziert.

Der Satellit beherbergt ein Teleskop als einzige Nutzlast. Die Bodenauflösung in Form der Pixelgröße bestimmt die Auslegung des Teleskops. Die Wellenlänge bei der das Teleskop arbeitet, muss im Infrarotbereich liegen und hat ebenfalls eine Auswirkung auf die Auslegung des Teleskops. Der Satellit soll für die Beobachtungen in einem niedrigen Erdorbit in 700km Höhe fliegen. Dies bedeutet, dass er die Erde in einem 90-Minuten Takt umkreist und die Inklination von 55° bestimmt die Überflughäufigkeit für die gewählte Zielregion, den nordamerikanischen Kontinent. Die Beobachtungsdaten werden an zwei Bodenstationen gesendet.

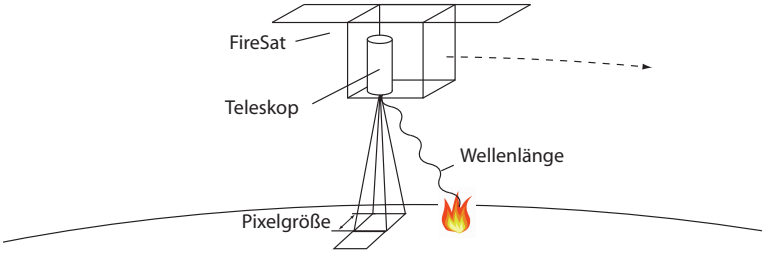


Abbildung 2.8: Skizze der Mission

2.3.2 Verwandte Arbeiten

Im Satellitenentwurf findet eine ständige Weiterentwicklung der Methoden, Tools und Prozesse statt. Unter dem Stichwort Model-Based Systems Engineering (MBSE) werden Methoden zur Formalisierung des Systems Engineering verstanden. Als Beispiel in der europäischen Raumfahrt sind die Bemühungen zur Standardisierung von Datenmodellen zur Speicherung [16] und zum Austausch von Daten [15] zu nennen. Diese Standards [11] werden in einem „Virtual Spacecraft Design“ (VSD) [18] genannten Projekt beispielhaft umgesetzt.

Neben diesen domänenspezifischen Datenmodellen gibt es Beispiele für die Modellierung von Systemen mit bereits vorhandenen Modellierungssprachen. Von einem INCOSE-Team wurde als Beispiel eine Modellierung des FireSat in der SysML durchgeführt [12, 13]. Des weiteren sind die Arbeiten von Karban [30] mit einem ausgedehnten SysML Modell und von Peak [51] mit der Integration von verschiedenen Domänenmodellen zu nennen. In Abgrenzung zu den Arbeiten über den FireSat mit SysML enthält diese Arbeit eine umfassende analytische Auslegung des Systems und aller Subsysteme. Im Kontrast zu Karban und Peak wird in dieser Arbeit der Entwurf nicht manuell, sondern maschinell über Regeln in einem durch die verschiedenen Klassendiagramme der Entwurfssprachen definierten Entwurfsraum aufgebaut. Damit werden die Möglichkeiten zur automatischen Exploration von Entwurfsräumen geschaffen.

Am besten vergleichen lässt sich die vorliegende Arbeit mit der Arbeit von Schaefer [63]. Sie ist eine direkte Vorgängerarbeit zu dieser Arbeit und ist ebenfalls in der Arbeitsgruppe Ähnlichkeitsmechanik am Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen der Universität Stuttgart entstanden. In [63] wurde bereits eine Satellitenentwurfssprache umgesetzt, die basierend auf analytischen Gleichungen eine automatische Auslegung des Gesamtsystems von ESEO (European Student Earth Orbiter) durchführen konnte. Als Weiterentwicklung zu [63] ist in dieser Arbeit die Geometrie nicht nur in Form „flacher“ Skripte enthalten, sondern es wird eine detaillierte Repräsentation der Geometrie in der UML vorgehalten, in der jedes geometrische Detail regelbasiert modifiziert werden kann. Als weitere Neuerung gegenüber [63] sind die Geometrie-, Thermal- und Regelungsmodelle, das integrierte automatische Packaging (Platzierung) und Routing (Verkabelung) sowie die automatisierte Untersuchung des Entwurfsraums zu nennen.

3 Aufbau von Entwurfssprachen

Die vorliegende Arbeit deckt den gesamten Satellitenentwurfsprozess ab. Durch die Methode der Entwurfssprachen ergibt sich für den Entwurfsprozess eine zum klassischen Entwurfsprozess unterschiedliche Aufgabenfolge. Diese Aufgabenfolge beginnt mit der Beschreibung des Problems in den Entwurfssprachen, also der Modellierung. Daher wird zunächst die Vorgehensweise für den Aufbau verschiedener Entwurfssprachen, die einen Satellitenentwurf abbilden, genauer erläutert.

Aufgrund der Kopplungen innerhalb des Entwurfsgebietes ergeben sich nach der Beschreibung in den Entwurfssprachen zwei Abfolgen des Entwurfsprozesses auf unterschiedlichen Ebenen. Auf abstrakter Ebene kommt nach der Vorauslegung die geometrische Konfiguration gefolgt von der physikalischen Integration des Systems. Auf detaillierter Ebene ergibt sich ein Entwurfszyklus, der auch die satellitenspezifischen Abhängigkeiten zwischen den einzelnen Subsystemen berücksichtigt.

3.1 Modellierung

Der Satellitenentwurf wird von der kontinuierlichen Modellierung des Entwurfs auf verschiedenen Abstraktionsebenen begleitet. Während der Entwicklung steigt der Detailgrad der Modellierung an. Für die Erstellung der Modelle werden in den Fachgebieten des Ingenieurwesens spezialisierte Computerprogramme verwendet. Im Vorentwurf werden oft grobe analytische Modelle oder Statistiken vergangener Entwürfe verwendet. Diese können mit Tabellenkalkulationsprogrammen wie Excel oder für weitergehende Analysen mit Computer Algebra Systemen wie Mathematica [71] oder Matlab [42] untersucht werden. Für den Detailentwurf wird eine geometrische Modellierung in CAD-Programmen wie CATIA V5 [10] oder für die thermale Modellierung werden Simulationsprogramme wie ESATAN-TMS [29] verwendet.

Im klassischen Entwurfsprozess werden in diesen Anwendungsprogrammen Modelle manuell aufgebaut. Beim Aufbau der verschiedenen Modelle werden oft Arbeiten doppelt ausgeführt und Informationen mehrfach eingegeben. Der Aufbau einer Komponente in einem CAD-Programm und in einem Thermalprogramm enthält zum Beispiel einen hohen Anteil an redundanten Arbeitsschritten. Um diesem Problem zu begegnen, wurden Austauschformate zwischen den Anwendungsprogrammen entwickelt (z.B. [4, 67]). Durch die Austauschformate können Modelle zwischen den Anwendungsprogrammen ausgetauscht werden. Dies führt zu erheblicher Reduzierung von Doppelarbeit. In Abb. 3.1 ist dieser Ansatz zum Datenaustausch über die Schnittstellen der Anwendungsprogramme auf der linken Seite schematisch dargestellt.

Der Datenaustausch auf diesem Weg beinhaltet jedoch, wie in [56] beschrieben, ein Problem. Durch die unterschiedliche Anzahl an Parametern in den Anwendungsprogrammen können

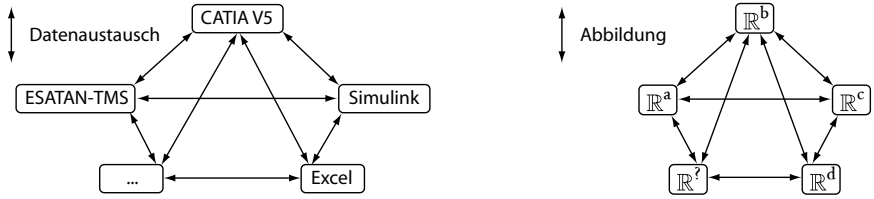


Abbildung 3.1: Vielfalt von Modellen nach [56]

deren Modelle nicht umkehrbar eindeutig aufeinander abgebildet werden. Betrachtet man jedes Modell als mathematischen Raum und jeden Parameter eines Modells als eine Dimension dieses Raums, so lässt sich das Problem mathematisch durch die fehlende bijektive Abbildung zwischen Räumen unterschiedlicher Dimensionalität beschreiben. Dies ist in Abb. 3.1 auf der rechten Seite dargestellt. Da ein CAD-Modell einen anderen Parametersatz als ein Thermalmodell hat, kann keines der beiden Modelle vollständig aus dem anderen erstellt werden. Daher fehlen Informationen beim Datenaustausch zwischen Anwendungsprogrammen und der Export von einem Programm in ein anderes ist zwingend mit manuellen Ergänzungen verbunden.

Der manuelle Aufbau der Modelle in den Anwendungsprogrammen beinhaltet einen großen Anteil von sich wiederholenden Routinetätigkeiten, wie zum Beispiel das Anbringen von Halterungen an Einbaugeräten. Um solche Routinetätigkeiten zu automatisieren, werden in Templates und Skripten Routineprozeduren festgelegt. Diese Prozeduren sind jedoch auf die Informationen des jeweiligen Domänenmodells beschränkt. Damit können keine weitergehenden funktionalen Aspekte miteinbezogen werden. Eine Halterung ist in einem CAD-Modell semantisch nur ein übliches Bauteil, bei welchem höchstens der Name einen Rückschluss auf die Funktion erlaubt. Damit können keine Operationen, die z.B. „alle Halterungen“ betreffen, umgesetzt werden.

Bei Änderungen im Verlauf des Entwurfsprozesses werden die Modelle in den verschiedenen Anwendungsprogrammen ebenfalls manuell verändert. Da die Modelle über den abgebildeten Satellitenentwurf miteinander gekoppelt sind, entsteht ein Konsistenzproblem zwischen den Modellen der verschiedenen Anwendungsprogramme. So müssen zum Beispiel die Maße einer Box im CAD-Modell immer mit den Maßen derselben Box im Thermalmodell übereinstimmen.

Um das Konsistenzproblem mit einem zentralen UML-Modell zu lösen, wurden in [52] verschiedene Modellierungstechniken und dazugehörige Schnittstellen entwickelt. Dabei wurde ein zentrales Modell aufgebaut und die Anwendungsdaten wurden in diesem Modell verknüpft. In Abb. 3.2 ist dies schematisch dargestellt. Dieser Ansatz hat zwei Vorteile. Erstens wird die Anzahl der Schnittstellen auf die Schnittstelle zum zentralen Modell reduziert. Zweitens wird das grundlegende Problem der unterschiedlichen Dimensionalität in den Modellen der Anwendungsprogramme durch ein zentrales Modell, das alle relevanten Daten enthält, aufgelöst.

Da im Produktentwurf oft auf vorhandene Komponenten zurückgegriffen wird, sollten auch deren Modelle wiederverwendbar gestaltet werden. Die Wiederverwendbarkeit der Modelle beschränkt sich jedoch bisher auf die Wiederverwendbarkeit von Modulen innerhalb von

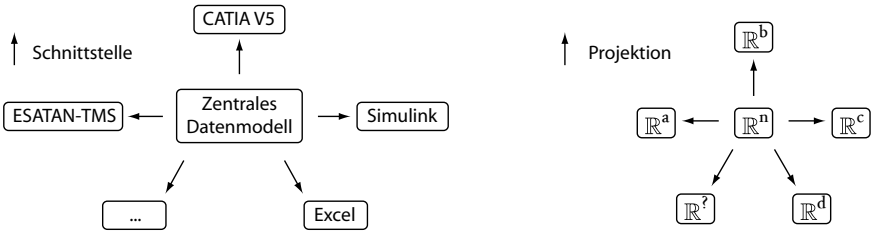


Abbildung 3.2: Lösung zentrales Modell nach [56]

Anwendungsprogrammen. So können zum Beispiel in CATIA Komponenten (Bauteile und Baugruppen) oder in Simulink Blocks (Simulationsblöcke) wiederverendet werden. Aus dieser Beschreibung lassen sich vier, in ihren Auswirkungen auf die Entwurfskosten oft unterschätzte Probleme des klassischen Entwurfsprozesses identifizieren:

- Die Beseitigung von *Doppelarbeit*,
- die Automatisierung von *Routinetätigkeiten*,
- das Auflösung des *Konsistenzproblems* und
- die Verbesserung der *Wiederverwendbarkeit*.

In dieser Arbeit wird erstmals für den Satellitenentwurf eine umfassende Behandlung dieser vier Probleme anhand der in [56] vorgeschlagenen Lösung mit einem zentralen Modell in Form von Entwurfssprachen umgesetzt. Die Lösung teilt sich auf in die Dekomposition des Entwurfsgebietes mittels der Vokabeln und in die Rekombination des Entwurfs mittels Regeln.

3.1.1 Dekomposition

Die Dekomposition in die verschiedenen Entwurfssprachen trägt zur Lösung der oben genannten Probleme bei. Wie die Dekomposition dafür gestaltet werden muss, wird nun im Einzelnen erläutert:

Doppelarbeit. Die Vermeidung der Doppelarbeit ist genau dann möglich, wenn zwei Arbeitsschritte, die auf der detaillierten Ebene unterschiedlich sind, auf einer abstrakteren Ebene identisch zusammenfallen. So ist zum Beispiel die Definition einer Box in einem CAD-Modell und einem Thermalmodell abstrakt die gleiche Tätigkeit. Der Unterschied zwischen einem Volumenmodell im CAD und einem Oberflächenmodell im Thermalmodell wird durch den Übersetzungsschritt beseitigt. Auf diese Weise hat man die Arbeit des Ingenieurs, der beide Modelle aufbauen muss, auf einen Compiler, der die Übersetzungsarbeit erledigt, verlagert. Der Ingenieur muss nur noch die Definition einer Box im abstrakten Modell vornehmen. Diese Abstraktion durch eine Entwurfssprache ist in Abb. 3.3 schematisch dargestellt.

Die Hierarchie in Abb. 3.3 links innerhalb der Entwurfssprachen ergibt sich aus folgender Überlegung: Jeder Satellit hat thermale Eigenschaften und jedes Thermalmodell hat geome-

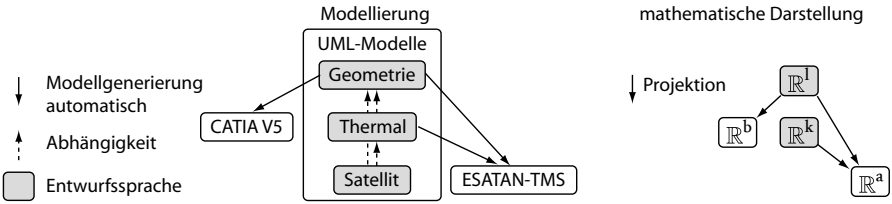


Abbildung 3.3: Abstraktionsebenen innerhalb Entwurfssprachen

trische Eigenschaften. Es muss hingegen nicht jedes geometrische Element thermale Eigenschaften haben und jedes thermale Element muss nicht Teil eines Satelliten sein. Deswegen ist die Entwurfssprache für Geometrie nicht von der Thermal-Entwurfssprache abhängig, sondern nur umgekehrt. Die Satelliten-Entwurfssprache ist hingegen von den beiden abstrakteren Entwurfssprachen abhängig.

Für die Abbildung von Geometrie in einer Entwurfssprache ist es notwendig, eine abstrakte Repräsentation der Geometrie in UML zu ermöglichen. Dies wurde in einer vom Autor betreuten Diplomarbeit von Jens Schmidt [65] umgesetzt. Mit dieser Entwurfssprache ist eine abstrakte Definition von Geometrie innerhalb eines UML-Modells möglich. Für die Modellierung der thermalen Eigenschaften wurde ebenfalls eine Entwurfssprache entwickelt. In der ebenfalls vom Autor betreuten Diplomarbeit von Muhammed Kocak [32] wurde diese Entwurfssprache entwickelt.

Konsistenz. Durch die verschiedenen Abstraktionsebenen kann die Geometrierepräsentation in verschiedene CAD-Tools exportiert werden. In dieser Arbeit werden die beiden CAD-Kerne von OpenCascade [49] und CATIA V5 [10] verwendet. Des weiteren kann die Geometrie auch in allen Simulationsmodellen, die Geometrie beinhalten, genutzt werden. Durch diesen Mechanismus lassen sich die Probleme der Doppelarbeit und der Konsistenz lösen. Da die Repräsentation im zentralen Modell aufgebaut wird, wird jeder Arbeitsschritt nur einmal ausgeführt. Da die Modelle aus der „single source of truth“, also aus *einem* Modell generiert werden und die Eigenschaften dort durch die hierarchische Ordnung nur an einer Stelle modelliert werden, kann die Konsistenz der Modelle sichergestellt werden.

Routinetätigkeiten. Die Automatisierung von Routinetätigkeiten wird in einer Entwurfssprache durch zwei Aspekte unterstützt. Zum einen sind in dem zentralen Modell einer Entwurfssprache alle relevanten Informationen des Entwurfs vorhanden. Dadurch können z.B. funktionale, operationelle und geometrische Informationen in den Routineprozeduren verwendet werden. Zum anderen gibt es für die Umsetzung von Prozeduren die Möglichkeit sowohl den Zeitpunkt im Entwurfsablauf als auch die Abstraktionsebene zu wählen. Da in einer Entwurfssprache der gesamte Entwurfsprozess abgebildet ist, kann der „richtige“ Zeitpunkt für die Ausführung einer bestimmten (zu automatisierenden) Tätigkeit frei gewählt werden.

In dem FireSat-Beispiel in dieser Arbeit werden die Kabel in Kapitel 4.5 erst nach Aufbau der Taschen eingebaut, um die Veränderungen in der Struktur beim Verlegen der Kabel zu berücksichtigen. Da die Entwurfsinformationen in einer hierarchischen Ordnung vorliegen, kann das Abstraktionsniveau, auf dem eine Tätigkeit ausgeführt wird, frei gewählt werden. So werden zum Beispiel die Halterungen in Abschnitt 4.4.2 an alle *MountedBoxes* angebaut und nicht an eine spezielle Box aus den Subsystem-Entwurfssprachen.

In Abb. 3.4 ist die Hierarchie der verschiedenen Entwurfssprachen für die Detaillierung des Geometriemodells abgebildet. Die Routing-Entwurfssprache wird in der Arbeit von Marc

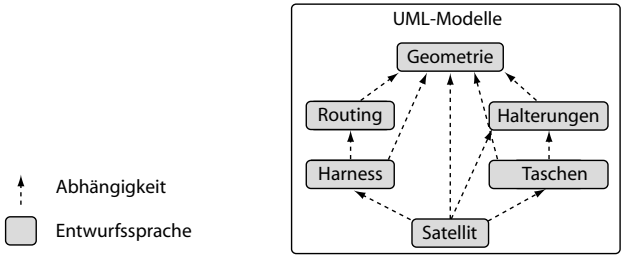


Abbildung 3.4: Hierarchie verschiedener Entwurfssprachen

Eheim [17] entwickelt. Die anderen Entwurfssprachen wurden im Rahmen dieser Arbeit aufgebaut. Durch die abgebildete Hierarchie ist es möglich, die Routing-Entwurfssprache für jeden Entwurf zu verwenden, der auf der Geometrie-Entwurfssprache aufbaut. Die speziellere Harness-Entwurfssprache kann nur verwendet werden, wenn auch die Routing-Entwurfssprache verwendet wird. Ebenso verhält es sich mit den Taschen und den Halterungen. Die Satelliten-Entwurfssprache hängt von allen gezeigten Entwurfssprachen ab.

Wiederverwendbarkeit. Für die Wiederverwendbarkeit von Komponenten werden die einzelnen Subsysteme des Satelliten als unabhängig voneinander verwendbare Entwurfssprachen entwickelt. Die Abhängigkeiten zwischen den verschiedenen Subsystemen untereinander und zum Gesamtsystem werden über eine Entwurfssprache für Satellitensysteme abgebildet. In dieser Arbeit wird die FireSat Mission als Beispiel verwendet. Die FireSat-Entwurfssprache ist in Abb. 3.5 ganz unten dargestellt.

Die Entwurfssprache für Satellitensysteme beinhaltet hauptsächlich die Systembudgets des Satelliten. In diesen Systembudgets werden die Eigenschaften und Anforderungen der verschiedenen Subsysteme und der Mission gesammelt und zusammengefasst. Die verschiedenen Subsysteme, wie zum Beispiel die Energieversorgung, erfüllen die in den Systembudgets abgebildeten Anforderungen. Dadurch wird das Subsystem über das Systembudget an das Gesamtsystem gekoppelt. Die Subsysteme bleiben dabei austauschbar und können für andere Missionen wiederverwendet werden.

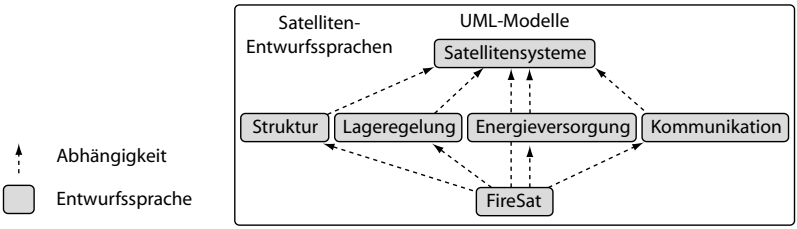


Abbildung 3.5: Hierarchie der Satelliten-Entwurfssprachen

In der FireSat-Entwurfssprache ganz unten in Abb. 3.5 werden alle missionsspezifischen Anpassungen am Entwurf vorgenommen. Wenn zum Beispiel bestimmte Komponenten eine vorgegebene Einbauposition erfordern, so kann dies in dieser Entwurfssprache definiert werden. Die Positionierung des Teleskops wird zum Beispiel in dieser Entwurfssprache vorgenommen. In der Hauptaktivität der FireSat-Entwurfssprache ist der gesamte Entwursprozess und damit die Reihenfolge, in der die Subsysteme eingebaut werden, definiert.

Ein Vorteil der Wiederverwendbarkeit mit Entwurfssprachen liegt in der automatischen Integration. Durch die zentrale Abbildung aller unterschiedlichen Modellbildungen für jede Komponente, können die wiederverwendeten Komponenten automatisch in allen Modellen gleichzeitig integriert werden. In den Vokabeln der Entwurfssprache sind für jede Komponente die gezeigten analytischen Auslegungsgleichungen sowie die später dargestellten Geometrie-, Thermal- und Regelungsmodelle enthalten.

3.1.2 Rekombination

Neben der zentralen Modellbildung mittels Vokabeln kann man in einer Entwurfssprache mithilfe von Regeln auch die Rekombination der Komponenten zu einem System abbilden. Nach der Methodik des systematischen Entwerfens von Pahl und Beitz [50] wird die Modellierung in unterschiedliche Kategorien unterteilt. Diese Kategorien sind in die Anforderungs-, Funktions-, Prinzip- und Gestaltmodellierung zu unterscheiden. In Abb. 3.6 sind diese vier Kategorien schematisch dargestellt. Nach diesem Schema werden die Anforderungen auf abstrakte Produktfunktionen übertragen und die Produktfunktionen werden durch Lösungsprinzipien umgesetzt. Die Komponenten werden dann entsprechend der Lösungsprinzipien ausgelegt. Diese vier verschiedenen Kategorien der Modellierung können in einer Entwurfssprache nebeneinander verwendet werden. Die Übertragungen der Anforderungen auf die Funktionen, der Funktionen auf die Prinziplösungen und der Prinziplösungen auf die Komponenten finden in einer Entwurfssprache durch Regeln statt.

Bei der Rekombination kann man zwei Vorgehensweisen unterscheiden, das Top-Down Vorgehen und das Bottom-Up Vorgehen. Beim Top-Down Vorgehen durchläuft der Entwurf die Kategorien in Abb. 3.6 strikt von links nach rechts. Beim Bottom-Up Vorgehen werden hingegen die Kategorien von rechts nach links durchlaufen.

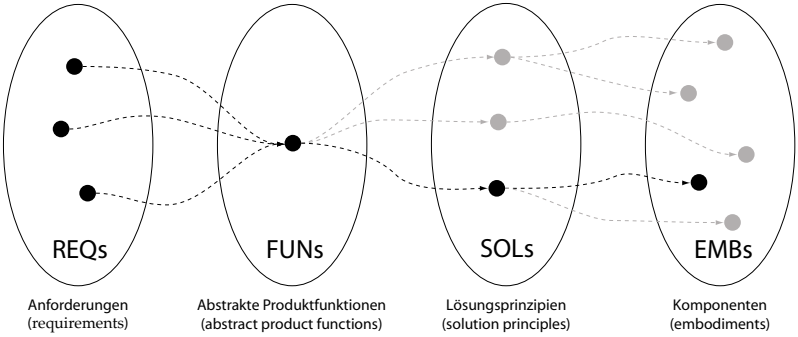


Abbildung 3.6: Modellierungskategorien nach [50]

Top-Down. Beim Top-Down Vorgehen ist bekannt, mit welchen Lösungsprinzipien eine Funktion erfüllt werden kann. Der Entwurfsprozess besteht aus der Auswahl des besten Lösungsprinzips und der Auswahl der besten Komponente unter den gegebenen Randbedingungen. Das Schema aus Abb. 3.6 wird dabei, wie in Abb. 3.7 dargestellt, von links nach rechts durchlaufen. Damit verläuft der Entwurfsprozess vom Abstrakten zum Konkreten hin.

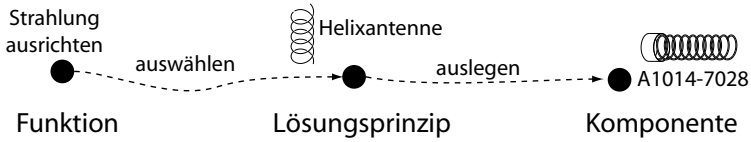


Abbildung 3.7: Top-Down Entwurf

Wenn die Funktionsforderung zum Beispiel lautet: „Strahlung ausrichten“, so kann man verschiedene Lösungsprinzipien von Antennen aus dieser Funktion ableiten. Diese Lösungsprinzipien, zum Beispiel eine Helixantenne, können dann von unterschiedlichen Komponenten, zum Beispiel der Antenne „A1014-7028“, umgesetzt werden. Auf diese Weise gelangt man von der abstrakten Funktion zu einer konkreten Komponente.

Bottom-Up. Beim Bottom-Up Vorgehen sind die Lösungsprinzipien für die Erfüllung einer Funktion oder die Komponenten, die ein Lösungsprinzip umsetzen, nicht bekannt. Der Entwurfsprozess besteht aus einem schrittweisen Aufbau der Struktur und die Erfüllung der funktionalen Anforderungen kann erst im Nachhinein überprüft werden. Es wird aus einer konkreten Handlungsanweisung die abstrakte Funktionsanforderung erfüllt.

Dies ist zum Beispiel beim Anbringen von Versteifungstaschen der Fall. In Abb. 3.8 ist rechts eine Platte mit Taschen dargestellt. Das Lösungsprinzip „Versickte Platte“ ist zwar

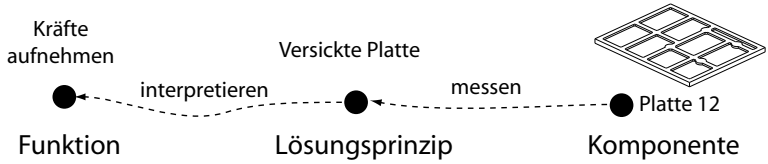


Abbildung 3.8: Bottom-Up Entwurf

bekannt, aber der Weg vom Lösungsprinzip zu einer richtigen Platte kann nicht Top-Down vorgegeben werden. Die Steifigkeit und Masse der Platte sind vielmehr das Resultat eines Konstruktionsvorgangs und können erst im Nachhinein gemessen werden. Daher erfolgt der Entwurf beim Bottom-Up Vorgehen von rechts nach links. Erst wird die Komponente nach Regeln aufgebaut, dann werden die relevanten Werte gemessen und auf das Lösungsprinzip übertragen. Ob die Funktion erfüllt wird, kann also erst nach Auslegen der Komponente interpretiert werden.

3.2 Problemgebiete

Durch die Abbildung der beiden Rekombinationsprinzipien in einer Entwurfssprache steht eine weitreichende Methode zur Beschreibung von Entwurfshandlungen zur Verfügung. Die einzelnen Entwurfshandlungen können jedoch nicht in einer beliebigen Reihenfolge ausgeführt werden. Durch die Abhängigkeiten innerhalb des entworfenen Systems ergibt sich eine logische Abfolge der Entwurfshandlungen. Nach dem beschriebenen Abbildungsproblem lassen sich im Satellitenentwurf nach [57] drei weitere Problemgebiete identifizieren. In Abb. 3.9 sind die drei Problemgebiete mit der vorhergehenden Beschreibung in einer Entwurfssprache dargestellt.

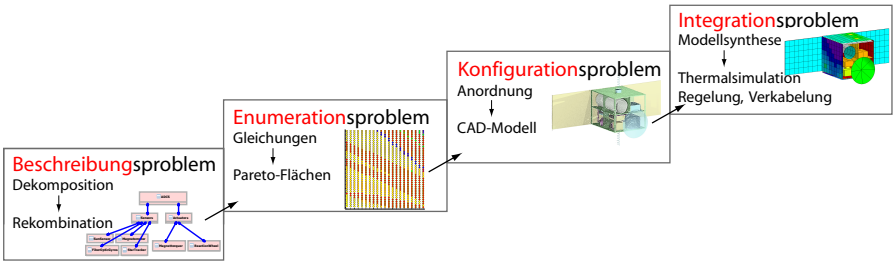


Abbildung 3.9: Problemgebiete beim Entwurf mit Entwurfssprachen nach [57]

Diese Problemgebiete werden in der Entwurfssprache für den FireSat-Satelliten von links unten nach rechts oben durchlaufen. Die Abfolge der Problemgebiete basiert auf der Top-Down Vorgehensweise in diesem Beispiel des Satellitenentwurfs. Die auf Analysen basierende

Auswahl aus den vielen möglichen Kombinationen aus Komponenten wird als Enumerationsproblem bezeichnet. Die Abfolge der verschiedenen Problemgebiete lässt sich durch folgende Abhängigkeiten erklären: Eine Auswahl zwischen den Komponenten kann erst dann stattfinden, wenn die Komponenten beschrieben sind. Die Konfiguration des Systems kann erst erfolgen, wenn bekannt ist, welche Komponenten Teil des Systems sind. Die physikalische Integration der Komponenten kann erst gelöst werden, wenn die Konfiguration des Systems bekannt ist. Daher können die Problemgebiete in Abb. 3.9 nur in dieser Reihenfolge durchlaufen werden.

Enumerationsproblem. Das Enumerationsproblem besteht in der Synthese und Analyse der verschiedenen Systemtopologien, um die beste unter ihnen zu finden. In einer Entwurfssprache erfolgt die Synthese der Komponenten zu einem System automatisch. Durch die Ausführung der Regeln werden die Vokabeln der verschiedenen Komponenten miteinander verbunden. Durch den Lösungspfadgenerator wird ein Gleichungssystem des Gesamtentwurfs erstellt und gelöst.

Die Analyse der verschiedenen Systemtopologien kann durch Auftragen der Eigenschaften über den Anforderungen vorgenommen werden. Dadurch entstehen Pareto-Flächen (wie z.B. in Abb. 4.19 dargestellt) anhand derer die verschiedenen Anforderungen gegeneinander abgewogen werden können. Auf Basis solcher Pareto-Flächen ist es denkbar, auch die Entscheidungen zwischen verschiedenen Systemtopologien zu automatisieren.

Konfigurationsproblem. Das Konfigurationsproblem besteht in der Anordnung der Komponenten zu einem System. Die Komponenten werden in der Satellitenstruktur auf Einbauebenen zugeordnet. Danach werden sie auf den Einbauebenen nebeneinander angeordnet. Die Anordnung kann auch automatisch mit einem Algorithmus geschehen.

Ein solcher Packaging-Algorithmus ist in dieser Arbeit aufbauend auf Vorarbeiten durch Marc Eheim [17] testweise umgesetzt. Er besteht aus einer schnellen Überprüfung der Überschneidungen zwischen den Bauteilen und einer daraus gefolgerten Bewegungsrichtung bei Überschneidung. Als Randbedingung können beliebige Freiheitsgrade der Komponenten bei der Bewegung eingeschränkt werden. Der Algorithmus funktioniert für Beispiele mit ausreichend Platz recht gut, jedoch nicht für die engeren Varianten des FireSat. Daher werden in dieser Arbeit parametrische Regeln verwendet, um verschiedene Systemtopologien anzuordnen. Nachdem das Konfigurationsproblem gelöst ist, gibt es ein vollständiges Geometriemodell des Satelliten. Damit können die Fragestellungen, deren Grundlage das Geometriemodell ist, beantwortet werden. Diese Fragestellungen beschreiben das Integrationsproblem.

Integrationsproblem. Bei den Integrationsproblemen muss die Funktionalität des Systems unter den gegebenen physikalischen Randbedingungen überprüft werden. Für die Behandlung der Integrationsprobleme werden aus der Entwurfssprache Modelle in den verschiedenen Anwendungsprogrammen generiert. In dieser Arbeit werden Modelle in CATIA V5, OpenCascade, ESATAN-TMS und Matlab-Simulink generiert. Zusätzlich findet eine Untersuchung der Verkabelung mittels der Routing-Entwurfssprache statt. Anhand dieser Modelle soll gezeigt

werden, wie im Entwurfsprozess mit Entwurfssprachen die Erstellung detaillierter Modelle für die Simulation vereinfacht wird.

Bei der Abfolge der Problemgebiete in Abb. 3.9 kann es jedoch unterschiedlich starke Rückkopplungen zwischen den Gebieten geben. Aus diesen Rückkopplungen ergibt sich bei genauerer Betrachtung des Entwurfsprozesses eine zyklisch verlaufende Entwurfssequenz.

3.3 Entwurfssequenz Satellit

Die detailliertere Entwurfssequenz für den Satelliten in diesem Beispiel ergibt sich durch die Abhängigkeiten innerhalb des Satellitenentwurfs. Durch die Gleichungen in den Vokabeln werden Abhängigkeiten formuliert. Aus diesen Abhängigkeiten lässt sich eine Reihenfolge der Gleichungslösung ableiten. Da die Vokabeln der Entwurfssprache durch die Regeln eingebaut werden, wirkt sich die Reihenfolge der Gleichungslösung auch auf die Reihenfolge der Regeln aus.

Deswegen kann aus den Gleichungen in diesem Entwurfsbeispiel eine Entwurfssequenz für den FireSat abgeleitet werden. Diese Darstellungsform findet sich bereits in den Arbeiten von Schaefer [62] und hat sich hier für den FireSat erneut bewährt. Die für den FireSat notwendige Reihenfolge der Regeln führt zu dem Entwurfszyklus in Abb. 3.10.

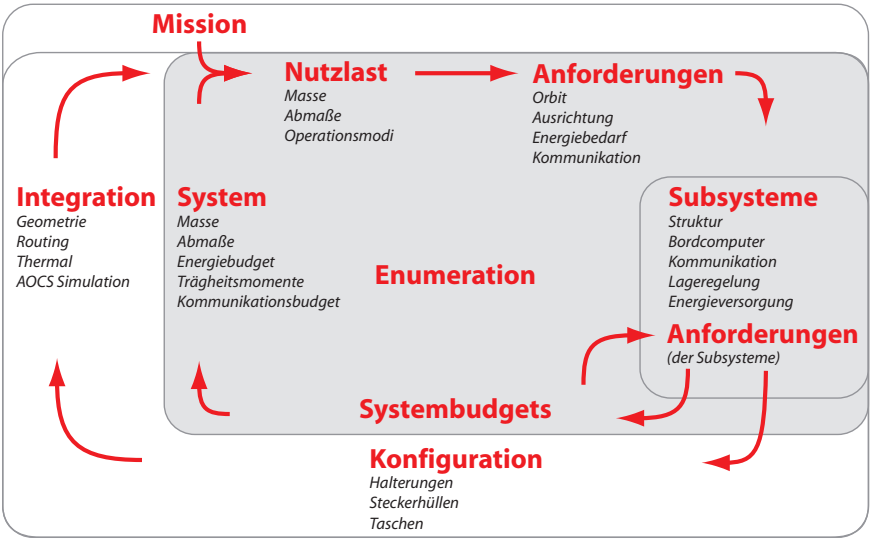


Abbildung 3.10: Entwurfszyklus für das FireSat-Beispiel [62]

Am Anfang eines Satellitenentwurfs steht die Missionsanalyse. In dieser Arbeit wird die Beispielmission FireSat aus [70] übernommen, daher wird die Missionsanalyse nicht weiter betrachtet. Von der Mission wird eine Nutzlast abgeleitet. Durch die funktionale Anforderung

einer Erdbeobachtung im Infrarotbereich ergibt sich ein Teleskop als Nutzlast. Dieses Teleskop bringt Seiteneffekte wie Masse und eine Baugröße sowie verschiedene Operationsmodi mit sich. Wenn das Teleskop ausgewählt ist, können weitere Anforderungen an das System abgeleitet werden. Dazu zählt zum Beispiel der Orbit, der über Regeln festgelegt wird. Die Ausrichtung der Nutzlast während der Beobachtung stellt eine entscheidende Anforderung an die Lageregelung des Satelliten dar. Der Energiebedarf der Nutzlast fordert den Einbau eines Energieversorgungssystems und die Anforderungen zur Datenübertragung an eine Bodenstation führen zum Einbau eines Kommunikationssystems.

Diese Subsysteme werden nach dem Aufbau der Nutzlast und des Orbits aufgrund der Anforderungen aus den Systembudgets eingebaut. Zuvor wird jedoch die Struktur eingebaut, da alle anderen Systeme in diese Struktur integriert werden müssen. Dann wird das Kommunikationssystem eingebaut. Das Datenverarbeitungssystem wird als nächstes aufgrund einer Abschätzung der benötigten Datenmengen ausgelegt. Danach folgt der Einbau des Energieversorgungssystems. Nach dem Energieversorgungssystem wird das Lageregelungssystem eingebaut.

Da die Subsysteme jeweils erneute Anforderungen mit einbringen können, ist eine Überprüfung der Systembudgets innerhalb dieses Entwurfszyklus notwendig. Die Problembehandlung kann dabei zum Teil automatisiert werden. Als Beispiel dafür wird in der vorgestellten Entwurfssprache das Energieversorgungssystem nach Einbau der Lageregelung noch einmal ausgelegt, falls nicht genug Leistung zur Verfügung steht.

Nach dem Einbau der Subsysteme können die Systembudgets über das Gesamtsystem gebildet werden. Entspricht der Entwurf den Anforderungen nicht, so können in den bis dahin erfolgten Entwurfsschritten Änderungen angebracht werden. Sind alle Anforderungen an den Entwurf erfüllt, so kann das Geometriemodell des Gesamtsystems in der Konfiguration aufgebaut werden. Dafür wird das Anordnungsproblem (Packaging) gelöst und verschiedene Verfeinerungen des Geometriemodells werden vorgenommen.

Nachdem das Geometriemodell aufgebaut ist können die Integrationsprobleme gelöst werden. Dafür werden aus der Entwurfssprache ein CAD-Modell, ein Thermal-Modell und ein Simulationsmodell generiert. Für die Verlegung der Kabel (Routing) wird des weiteren die Entwurfssprache für Verkabelung von Marc Eheim [17] angesteuert.

Mit diesem automatischen Entwurfszyklus ist es möglich, innerhalb von wenigen Minuten[1] einen vollständigen Satellitenentwurf durchzuführen. Die Simulationen in den Anwendungsprogrammen können dann zum Teil längere Laufzeiten[2] in Anspruch nehmen. Der wichtigste Aspekt ist jedoch, dass die oben genannten Probleme des klassischen Entwurfsprozesses damit beseitigt sind und die Arbeit des Ingenieurs auf die eigentliche Entwurfsarbeit konzentriert wird. Im Folgenden soll nun gezeigt werden, wie dies ermöglicht wird und welche Resultate dies mit sich bringt.

[1] Übersetzung der Entwurfssprache 3min 11sec auf Intel Core2 Quad 2.5 Ghz mit 8 GB RAM.
[2] Export nach CATIA 6min 46sec, Simulation in Simulink 15min, Simulation in ESATAN 2-3h

4 Entwurfssprachen für Satelliten

In diesem Kapitel wird der Aufbau und Einsatz der verschiedenen Entwurfssprachen für den Satellitenentwurf erläutert. Um den Aufbau der Entwurfssprachen zu demonstrieren wird die Modellierung und die Enumeration in Abschnitt 4.1 am Beispiel des Kommunikationssystems auf Subsystem- und Komponentenebene erläutert. In Abschnitt 4.2 wird die Beschreibung der weiteren Subsysteme des Satelliten in verschiedenen Entwurfssprachen vorgestellt. Die Nutzlast für den gewählten Beispielsatelliten FireSat wird als letztes Subsystem beschrieben.

Die automatische Synthese des Gesamtsystems wird in Abschnitt 4.3 dargestellt. Durch die automatische Synthese ergeben sich umfangreiche Analysemöglichkeiten, die ausschnittsweise demonstriert werden. Nach der analytischen Auslegung des Gesamtsystems wird in Abschnitt 4.4 die Behandlung von Geometrie für die Konfiguration des Satelliten in einer Entwurfssprache erläutert. Die Integration eines komplexen Gesamtsystems wird dann in Abschnitt 4.5 am Beispiel des FireSat gezeigt.

4.1 Entwurfssprache Kommunikationssystem

In diesem Abschnitt wird eine Entwurfssprache für die Auslegung von Kommunikationssystemen für Satelliten beschrieben. Das Kommunikationssystem eines Satelliten ist für den Datenaustausch des Satelliten mit den Bodenstationen verantwortlich. In Abb. 4.1 ist eine Skizze des Systems dargestellt.

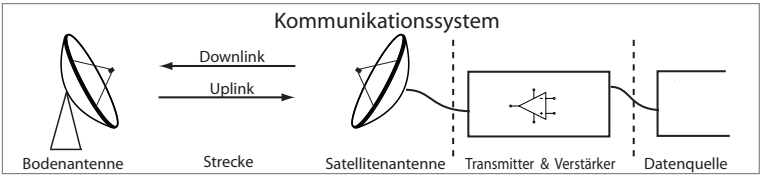


Abbildung 4.1: Skizze des Kommunikationssystems

Man unterscheidet die verschiedenen Übertragungsrichtungen (Up- und Downlink) sowie die Übertragung von Nutzlastdaten einerseits und Daten zur Überwachung und Steuerung des Satelliten (Telemetrie und Telecommand) andererseits. In diesem Beispiel soll die Auslegung für den Downlink der Nutzlastdaten behandelt werden.

In der Modellierung mit UML wird das System in Klassen unterteilt. Die Klassen werden mittels Attributen und Gleichungen beschrieben. Für den Zusammenbau des Systems aus

den Klassen werden Regeln definiert. Für die einzelnen Komponenten werden automatisch Gleichungssysteme aufgebaut und gelöst. Dadurch können die Komponenten miteinander verglichen werden. Für das zusammengesetzte System wird ebenfalls ein Gleichungssystem automatisch aufgestellt und gelöst. Dann können die verschiedenen Alternativen für das System miteinander verglichen und allgemeine Regeln für die Auswahl der Komponenten abgeleitet werden.

4.1.1 Modellierung

Die Klassen für das Kommunikationssystem orientieren sich an der Skizze in Abb. 4.1. Die Klasse für die Berechnung der Strecke *SatelliteLink* ist als zentrale Klasse des Systems in Abb. 4.2 dargestellt.

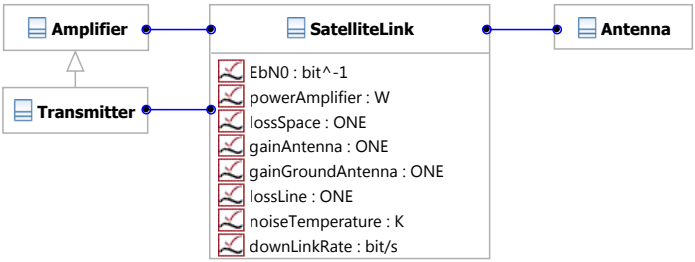


Abbildung 4.2: Klassen des Kommunikationssystems

In der *SatelliteLink*-Klasse ist die Gleichung (4.1) für die Berechnung der Übertragungsstrecke hinterlegt.

$$\frac{E_b}{N_0} = \frac{powerAmplifier \cdot gainAntenna \cdot gainGroundAntenna}{boltzmann \cdot noiseTemperature \cdot downLinkRate \cdot lossLine \cdot lossSpace} \tag{4.1}$$

Das Verhältnis E_b/N_0 - der Energie eines Bit zu der Rauschleistungsdichte - ist die Übertragungsqualität der Verbindung. Die Übertragungsqualität ergibt sich aus den Größen des Systementwurfs auf der rechten Seite der Gleichung. Das System besteht neben dem *SatelliteLink*, wie in Abb. 4.2 dargestellt, aus einem *Amplifier*, einem *Transmitter* und einer *Antenna*.

Die erste Variable auf der rechten Seite (`powerAmplifier`) ist die Verstärkerleistung des jeweiligen *Amplifier*. Der *Transmitter* wandelt die Daten für die Übertragung um und kann die Rolle eines *Amplifier* übernehmen. Es kann jedoch auch ein *Amplifier* getrennt vom *Transmitter* eingebaut werden. Der Antennengewinn (`gainAntenna`) wird von der jeweils eingesetzten *Antenna* berechnet.

Die restlichen Größen in Gleichung (4.1) sind im Zähler der Antennengewinn der Bodenantenne (`gainGroundAntenna`) und im Nenner die Boltzmann-Konstante (`boltzmann`),

die Rauschtemperatur des Systems (`noiseTemperature`), die geforderte Übertragungsrate (`downLinkRate`) und die Leitungsverluste an Bord des Satelliten (`lossLine`) sowie auf der Strecke (`lossSpace`).

Für die Auslegung des Systems wird neben der Übertragungsqualität und der Übertragungsrate die Systemmasse als Bewertungskriterium verwendet. Dafür werden die Massen der einzelnen Komponenten zusammengezählt. Die Eigenschaft `mass` wird dabei von den in Abschnitt 4.2.1 vorgestellten Systemklassen an das Kommunikationssystem vererbt. Dadurch ergibt sich Gleichung (4.2) mit den Instanznamen vor den Variablen, um die Zugehörigkeit der `mass`-Variablen zu verdeutlichen.

$$satelliteLink.mass = antenna.mass + transmitter.mass + amplifier.mass \tag{4.2}$$

Die Darstellung mit den Instanznamen entspricht der internen Verarbeitungsform jeder Gleichung in einer Entwurfssprache. In den folgenden Gleichungen dieses Kapitels werden nur die Namen der Variabeln verwendet, um die Lesbarkeit zu vereinfachen.

Um das System nach Gleichung (4.1) auslegen zu können, müssen die einzelnen Variablen berechnet werden. Um den Gewinn der *Antenna* (`gainAntenna`) berechnen zu können, ist es notwendig ein Lösungsprinzip für die Antenne auszuwählen. Für die Auslegung gibt es die Anforderung einer Wellenlänge (`lambda`), bei der das System arbeiten soll und die Anforderung der Masse (`mass`) der Antenne. Aus diesen Anforderungen ergibt sich das in Abb. 3.6 abstrakt dargestellte Schema nach Pahl und Beitz zu der Darstellung in Abb. 4.3.

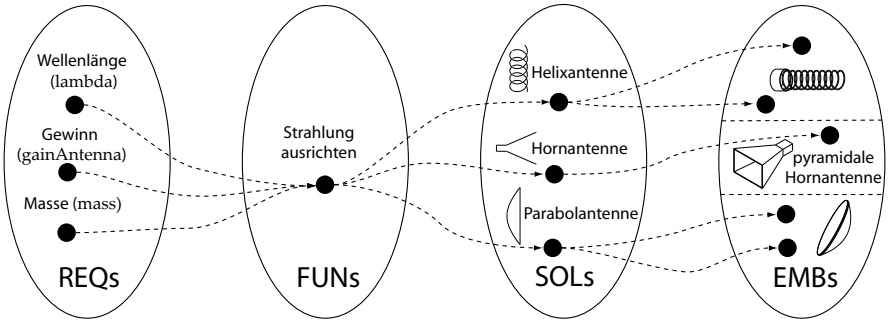


Abbildung 4.3: Antennen in Modellierungskategorien nach [50]

In Abb. 4.3 werden die Anforderungen (REQs) auf (hier nur) eine abstrakte Produktfunktion (FUNs) „Strahlung ausrichten“ übertragen. Diese Produktfunktion beschreibt die Funktionalität einer Antenne. Von der Produktfunktion sind verschiedene Abbildungen auf ein Lösungsprinzip (SOLs) möglich. Im Raum der Lösungsprinzipien gibt es drei Antennenarten zur Auswahl. Eine Helixantenne hat als Wirkprinzip einen helixförmigen Leiter, entlang dessen die Strahlung koaxial ausgerichtet wird. Eine Hornantenne hat als Wirkprinzip ein sich öffnendes Horn,

das als Hohlleiter fungiert. Eine Parabolantenne hat einen parabelförmigen Reflektor, der von einem Hornstrahler im Brennpunkt gespeist wird und die Strahlung reflektiert und ausrichtet.

Bei der Übertragung des Lösungsprinzips auf die konkreten Komponenten (EMBs) sind verschiedene Auslegungen der jeweiligen Antenne möglich. Eine Helixantenne kann zum Beispiel zwischen 7 und 35 Windungen haben. In Abb. 4.4 ist das Klassendiagramm zu den drei verschiedenen Antennen dargestellt.

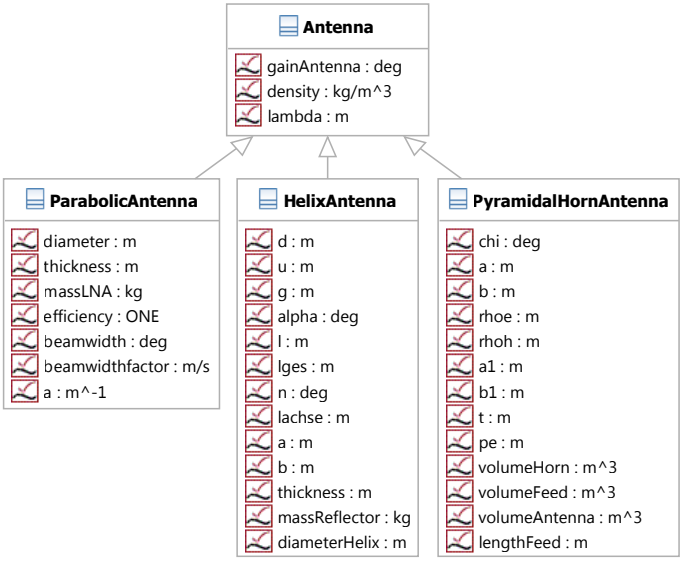


Abbildung 4.4: Antennen-Lösungsprinzipien mit Oberklasse

Die Oberklasse *Antenna* repräsentiert die Klasseneigenschaften der Funktion „Strahlung ausrichten“. Sie beinhaltet alle gemeinsamen Eigenschaften der Antennen. Die drei Unterklassen beschreiben die speziellen geometrischen Attribute der jeweiligen Lösungsprinzipien. In den Klassen sind auch die Auslegungsgleichungen der Antennen hinterlegt. Die Gleichungen drücken die Kopplungen zwischen den Anforderungen `lambda`, `gainAntenna` und `mass` aus.

Im Folgenden wird die Auslegung der einzelnen Antennen gezeigt. Als erstes wird eine Parabolantenne ausgelegt. Des weiteren werden eine pyramidale Hornantenne und eine Helixantenne ausgelegt. Für die bessere Lesbarkeit werden die griechischen Buchstaben in den Gleichungen als Symbole abgedruckt, im Klassendiagramm sind sie hingegen ausgeschrieben. Es gilt also:

`lambda`	=	λ,		`rhoe`	=	ρ_e,
`chi`	=	χ,		`rhoh`	=	ρ_h,
`alpha`	=	α	und	`lges`	=	l_{ges} etc.

ParabolicAntenna. Eine Parabolantenne ist eine Antenne mit einem parabelförmigen Spiegel (Abb. 4.5). Der Spiegel wird in diesem Beispiel direkt durch einen rauscharmen Verstärker (Low-Noise-Amplifier (LNA)) gespeist.

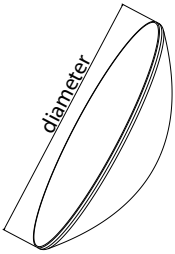


Abbildung 4.5: Skizze des Primärspiegels der Parabolantenne

Die Masse des Primärspiegels wird in Gleichung (4.3) in Abhängigkeit der Materialdicke (`thickness`), der Dichte (`density`), des Krümmungsparameters (`a`) und des Durchmessers (`diameter`) angegeben. Der Krümmungsparameter für diese Berechnungen ist a = 1/m, die Dichte ist 370 kg/m³ und die Dicke 0.0025 m. Diese Masse wird in den nachfolgenden Berechnungen näherungsweise als Masse (`mass`) der Antenne verwendet, da die Massen der Halterungen sowie des rauscharmen Verstärkers als konstant angesehen werden können.

In Gleichung (4.4) ist der Gewinn (`gainAntenna`) der Antenne als Funktion des Antennendurchmessers (`diameter`) sowie der Wellenlänge des Trägers (λ) angegeben.

$$\text{mass} = \frac{\pi}{a \cdot 6} \cdot \mathit{diameter} \cdot \left(1 + 4 \cdot a^2 \cdot \mathit{diameter}^2\right) \cdot \sqrt{4 + \frac{1}{a^2 \cdot \mathit{diameter}^2}} \cdot \mathit{thickness} \cdot \mathit{density} \tag{4.3}$$

$$\mathit{gainAntenna} = \pi^2 \cdot \frac{\mathit{diameter}^2}{\lambda^2} \tag{4.4}$$

PyramidalHornAntenna. Als zweite Alternative für das Kommunikationssystem wird eine Hornantenne nach [6, Seite 770 ff.] ausgelegt. Diese *PyramidalHornAntenna* besteht aus einem rechteckigen Hohlleiter mit einer pyramidalen Erweiterung am Ende.

Die einzelnen Werte für die Abmessungen und die beiden Öffnungswinkel des Horns können aus der Wellenlänge und dem benötigten Antennengewinn berechnet werden. Mit Gleichung (4.5) wird aus einem Näherungswert für χ, dem Verhältnis zwischen der Länge des Horns und λ, die Länge des Strahlers ρ_e berechnet. Mit einer Vorgabe für den zuführenden Hohlleiter von $0.7 \cdot \lambda$ Breite lassen sich alle anderen Geometrievariablen durch die Annahme eines

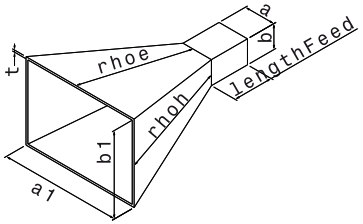


Abbildung 4.6: Skizze einer Hornantenne

Breite/Höhe Verhältnisses von 2/1 durch die Gleichungen (4.6 - 4.8) bestimmen. Die Masse wird über das Volumen mit Gleichung (4.9 - 4.11) bestimmt, die Dichte ist 2700 $\mathrm{kg/m^3}$ und die Dicke 0.0002 m.

$$\rho_e = \chi \cdot \lambda \quad \text{mit dem Anfangswert} \quad \chi = \frac{gainAntenna}{2 \cdot \pi \cdot \sqrt{2 \cdot \pi}} \tag{4.5}$$

$$\frac{\rho_h}{\lambda} = \frac{gainAntenna^2}{8 \cdot \pi^3} \cdot \frac{1}{\chi} \tag{4.6}$$

$$p_e = (b_1 - b) \cdot \sqrt{\frac{\rho_e{}^2}{b_1} - \frac{1}{4}} \tag{4.7}$$

$$a_1 = \sqrt{3 \cdot \lambda \cdot \rho_h} \quad \text{und} \quad b_1 = \sqrt{2 \cdot \lambda \cdot \rho_e} \tag{4.8}$$

$$\begin{aligned} volumeHorn = & \frac{p_e}{3} \cdot ((a_1 + t) \cdot (b_1 + t) + \\ & \sqrt{((a1 + t) \cdot (b_1 + t) \cdot (a + t) \cdot (b + t))} - \\ & (a + t) \cdot (b + t) + a_1 \cdot b_1 + a \cdot b + \sqrt{(a_1 \cdot b_1 \cdot a \cdot b)}) \end{aligned} \tag{4.9}$$

$$\begin{aligned} volumeFeed = & (a + 2 \cdot t) \cdot (b + 2 \cdot t) \cdot lengthFeed - \\ & a \cdot b \cdot lengthFeed \end{aligned} \tag{4.10}$$

$$mass = (volumeFeed + volumeHorn) \cdot density \tag{4.11}$$

HelixAntenna. Eine Helixantenne besitzt, wie in Abbildung 4.7 dargestellt, einen spiralförmig aufgewickelten Leitungsdraht, der am unteren Ende der Antenne meist koaxial gespeist wird. In der hier gewählten Auslegung ist am unteren Ende der Antenne ein reflektierender Schirm angebracht. Die Berechnungen werden nach [31, Seite 401 ff.] durchgeführt.

Der Gewinn der Antenne lässt sich in einem Bereich von 7 bis 35 Windungen mit Gleichung (4.12) näherungsweise bestimmen. Es wird ein Umfang von $u = \lambda$ gewählt. Die Größe des

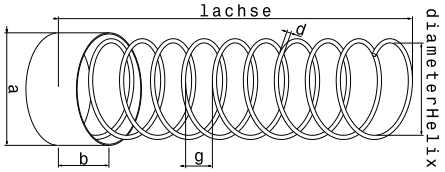


Abbildung 4.7: Skizze einer Helixantenne

Reflektors wird nach Gleichung (4.14) an die Wellenlänge gekoppelt.

$$gainAntenna = 8.3 \cdot \left(\frac{u}{\lambda}\right)^{\sqrt{n+2}-1} \cdot \left(\frac{n \cdot g}{\lambda}\right)^{\frac{4}{5}} \cdot \left(\frac{\tan(0.2182)}{\tan(\alpha)}\right)^{\frac{\sqrt{n}}{2}} \tag{4.12}$$

$$l^2 = u^2 + g^2 \quad \text{und} \quad l_{ges} = l \cdot n \tag{4.13}$$

$$a = 0.8 \cdot \lambda \quad \text{und} \quad b = \frac{a}{2} \tag{4.14}$$

Die Masse der Helixantenne setzt sich aus der Masse des Leitungsdrahts (Gleichung 4.15) und der Masse des Reflektors (Gleichung 4.16) zusammen:

$$mass = density \cdot d^2 \cdot \frac{\pi}{4} \cdot l_{ges} + massReflector \tag{4.15}$$

$$massReflector = \left(a^2 \cdot \frac{\pi}{4} \cdot thickness + b \cdot \pi \cdot a \cdot thickness\right) \cdot density \tag{4.16}$$

Transmitter und Verstärker. Neben der Antenne besteht das System aus einem Transmitter und einem Verstärker. Der *Transmitter* erbt wie in Abb. 4.8 dargestellt vom *Amplifier*. Dies bedeutet, dass er ebenfalls die Eigenschaften eines *Amplifier*s besitzt. Zusätzlich zu dem *Transmitter* kann eine Wanderfeldröhre (Travelling Wave Tube Amplifier (TWT) als *Amplifier*

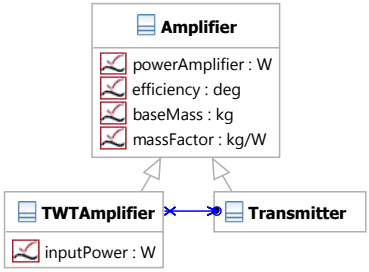


Abbildung 4.8: Transmitter- und Verstärker-Klassen

zum Einsatz kommen. In diesem Fall ist die Ausgangsleistung des *Transmitter* gleich der Eingangsleistung `inputPower` der Wanderfeldröhre. Um verschiedene Kombinationen von Komponenten zu vergleichen, ist es notwendig, die *Amplifier* in verschiedenen Größen auslegen zu können. Da es schwierig ist, eine allgemeine Auslegungsgleichung für beliebige Verstärker herzuleiten, wird für die Analyse zwischen verschiedenen existierenden Komponenten linear interpoliert. Daraus resultiert eine einfache Gleichung (4.17) für die Masse mit dem Leistungsbedarf nach (4.18).

$$mass = power \cdot massFactor + baseMass \tag{4.17}$$

$$power = \frac{powerAmplifier}{efficiency} \tag{4.18}$$

Für die Massenberechnung des Transmitters werden die Werte

$$massFactor = 0.008\ {}^{kg}/_{W}, \tag{4.19}$$

$$baseMass = 0.5\ \text{kg und} \tag{4.20}$$

$$efficiency = 0.1\ \text{nach [37]} \tag{4.21}$$

eingesetzt. Für die Berechnung der Wanderfeldröhre werden hingegen die Werte

$$massFactor = 0.005\ {}^{kg}/_{W}, \tag{4.22}$$

$$baseMass = 2\ \text{kg und} \tag{4.23}$$

$$efficiency = 0.7\ \text{nach [35]} \tag{4.24}$$

eingesetzt. Die höhere Effizienz der Wanderfeldröhre wird mit einer größeren Anfangsmasse erkauft. Damit kann man erwarten, dass es sich nur für höhere Leistungsbereiche lohnt, eine Wanderfeldröhre einzusetzen.

4.1.2 Auslegung Antennen

Mit der vorgestellten Modellierung der Antennen können nun die Gleichungen der Antennen ineinander eingesetzt und nach den Anforderungen aufgelöst werden. Mit den Gleichungen für die einzelnen Antennen lassen sich die Masse und der Gewinn der jeweiligen Antenne unter Angabe der Wellenlänge bestimmen. Die drei Anforderungen an die Produktfunktion

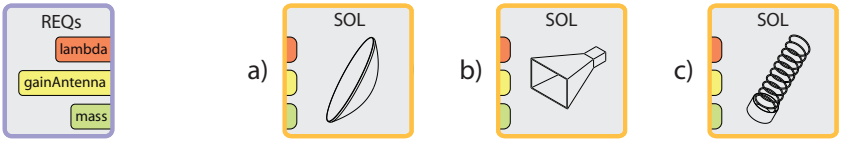


Abbildung 4.9: Anforderungen und Lösungsprinzipien

„Strahlung ausrichten“ können zunächst scheinbar unabhängig voneinander festgelegt werden. Diese Situation ist in Abb. 4.9 dargestellt.

Wählt man nun ein Lösungsprinzip aus, so werden die Anforderungen durch die Antennenvariablen gekoppelt. Diese Situation ist in Abb. 4.10 dargestellt. Die drei Anforderungen werden

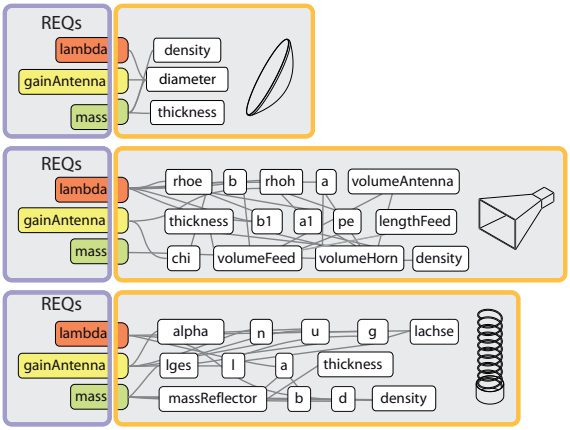


Abbildung 4.10: Kopplung der Anforderungen durch die Lösungsprinzipien

durch jedes der drei Lösungsprinzipien auf unterschiedliche Art und Weise gekoppelt. Wird eine Anforderung geändert, so entscheidet das Lösungsprinzip über den „Wechselkurs“ zwischen den Anforderungen. Dies ist eine wichtige Erkenntnis im Entwurf komplexer Systeme:

„Das Lösungsprinzip entscheidet wie die Anforderungen gekoppelt werden[1].“

Von den Variablen der Parabolantenne in Abb. 4.10 im obersten Bild ist noch nicht festgelegt, welche Werte vorgegeben werden und welche berechnet werden. Legt man die vorgegebenen Werte fest, so bekommt man einen Lösungspfad, entlang dessen die weiteren Variablen berechnet werden können. Dieser Lösungspfad lässt sich wie in Abb. 4.11 als Entwurfszyklus darstellen.

Dieser Entwurfszyklus gilt für die Auslegung der Antenne bei gegebener Gewinnforderung und gesuchter Masse. Die gestrichelten Rückpfeile stellen die Bewertungsmöglichkeiten im Entwurfsprozess dar. An diesen Stellen kann der Entwurf überprüft und es können Änderungen eingebracht werden.

Der Lösungspfad für die Helixantenne ist in Abb. 4.12 dargestellt. Bei der Helixantenne ist es nicht möglich, den Antennengewinn vorzugeben. Dieser muss aus einer Vorgabe der Windungsanzahl sowie der Steigung der Wendel berechnet werden.

[1] Unter „Lösungsprinzip“ wird in diesem Zusammenhang die mathematische Modellierung des Lösungsprinzips in einem Gleichungssystem verstanden.

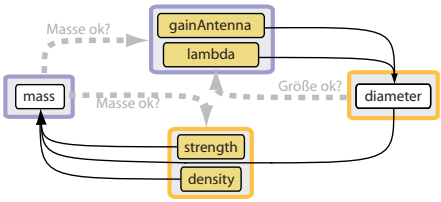


Abbildung 4.11: Entwurfszyklus der Parabolantenne

In Abbildung 4.12 ist zu sehen, dass der Gewinn erst nach der Berechnung der geometrischen Werte angegeben werden kann. In einem solchen Fall entsteht im Entwurf eine Iterationsschleife. Die geforderte Größe kann nicht direkt berechnet werden, sondern muss durch eine Iteration der Randbedingungen auf den richtigen Wertebereich erreicht werden.

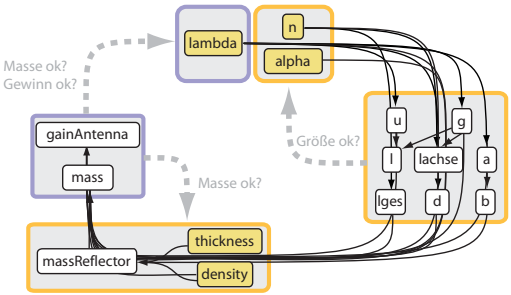


Abbildung 4.12: Entwurfszyklus der Helixantenne

Vergleich der Antennen. Nimmt man eine Wellenlänge von 12 cm an (Frequenz $\approx$ 2.5 Ghz), so kann man die Masse der drei Komponenten über den Gewinn auftragen. In Abb. 4.13 sind die Massen der Antennen in Kilogramm über dem Gewinn in Dezibel aufgetragen.

Man kann sehen, dass die Helixantenne eine Lösung für niedrige Gewinnwerte darstellt. Im Übergang zu höheren Gewinnwerten ist zunächst die Hornantenne die leichteste Lösung. Die Parabolantenne ist bei Gewinnwerten von 16 bis 19 dB noch schwerer als die Hornantenne. Bei einem Wert um 19 dB schneiden sich jedoch die beiden Funktionen der Antennen. Für eine Entwurfsentscheidung anhand der geringsten Masse findet an diesem Punkt ein Wechsel vom Lösungsprinzip „Hornantenne“ auf das Lösungsprinzip „Parabolantenne“ statt.

Auf diese Weise können Entwurfsentscheidungen automatisiert getroffen werden. Da die Entscheidung jedoch in solchen „Schaltpunkten“ stattfindet und eine minimale Abweichung bei Entwürfen in diesem Gebiet ein anderes Lösungsprinzip hervorrufen kann, stellt sich die Frage, wie stabil diese Entscheidung ist.

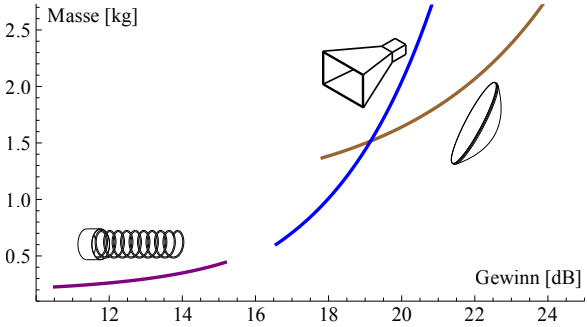


Abbildung 4.13: Masse der Lösungsprinzipien

Stabilität der Entwurfsentscheidungen. Um die Stabilität dieser Entwurfsentscheidung zu überprüfen, wird der Schnittpunkt zwischen den beiden Lösungsprinzipien genauer betrachtet. Da die Auslegung einer Komponente nach einem Lösungsprinzip eine Bandbreite an Möglichkeiten eröffnet, resultiert daraus eine Unschärfe für den Entscheidungspunkt.

Um einen qualitativen Unterschied in der Auslegung der Komponenten zu untersuchen, wird angenommen, dass eine bessere Komponente (Horn- oder Parabolreflektor) eine dünnere Wandstärke aufweist oder aus einem leichteren Material gefertigt ist. Trägt man eine kombinierte Schwankungsbreite dieser Eigenschaften von 10% für die Lösungsprinzipien[2] auf, so ergibt sich Abb. 4.14.

In dieser Abbildung sind vier markante Punkte P1 - P4 eingetragen. Bei Punkt P1 durchschneidet die untere Linie des Lösungsprinzips Parabolantenne die obere Linie des Lösungsprinzips Hornantenne. Dies bedeutet, dass ab diesem Gewinnwert eine „gute“ und damit eine leicht konstruierte Parabolantenne besser (d.h. leichter) ist als eine „schlechte“ und damit schwer konstruierte Hornantenne. Bei Punkt P2 wird die schwere Hornantenne auch von einer „schlechten“ Parabolantenne abgelöst. Bei Punkt P3 wird die „gute“ Hornantenne von der „guten“ Parabolantenne abgelöst. Ab Punkt P4 ist eine Parabolantenne in jedem Fall die leichteste Lösung.

Anhand der Form der Überschneidungsfläche in Abb. 4.14 können für diese Kurvenform und mit einer zum Mittelwert parallel verlaufenden Schwankungsbreite weitere Aussagen über die Stabilität einer Entwurfsentscheidung getroffen werden. Wenn die Steigungen der beiden Funktionen sehr unterschiedlich sind, schneiden sich die Kurven im Extremfall mit einem Winkel von 90°. In diesem Fall ist die Überschneidungsfläche am kleinsten und die Entscheidung zwischen den Prinziplösungen fällt in einen kleinen Wertebereich. Man könnte von einem „harten“ Entscheidungspunkt sprechen. Sind die Steigungen der beiden Funktionen

[2]Prinzipiell wäre hier eine statistische Betrachtung der Unschärfe der betrachteten Entwurfsvariablen angebracht, um die Überschneidung genauer zu analysieren. Hier soll jedoch nur der Gedanke eines Stabilitätskriteriums umrissen werden und deshalb ein Unschärfewert, ohne stastistische Begründung, angenommen werden.

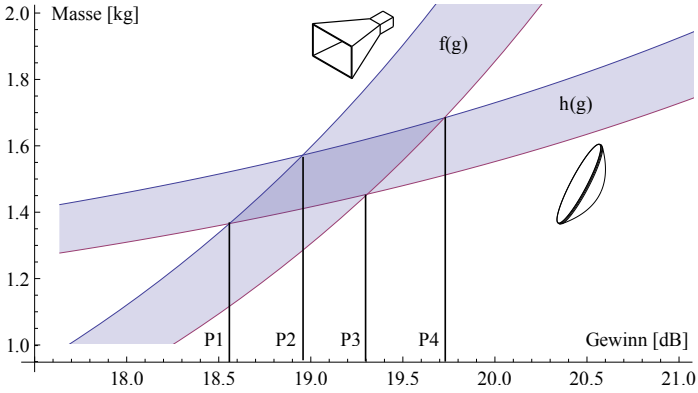


Abbildung 4.14: Schwankungsbreite der Masse der Lösungsprinzipien

hingegen sehr ähnlich im Schnittpunkt, so ergibt sich eine längere Überschneidungsfläche. Man könnte von einem „weichen“ Entscheidungspunkt sprechen.

4.1.3 Enumeration Kommunikationssystem

Die Auswahl der Komponenten für das System wird nach Abb. 3.9 als Enumeration bezeichnet. Für die Analyse des Kommunikationssystems müssen die Komponenten in das System integriert werden. Dafür muss neben der bereits oben gezeigten Auslegung der Antenne auch ein Transmitter und ein Verstärker ausgelegt werden. In diesem Beispiel wird die Masse dieser Komponenten nach Gleichung (4.17) und (4.18) über die vom System geforderte Verstärkerleistung berechnet.

Die Integration der Antenne in das Kommunikationssystem ist in Abb. 4.15 schematisch dargestellt. Durch den Wechsel von Komponentenebene auf Systemebene ergeben sich im Vergleich zu Abb. 4.9 neue Anforderungen. Die Antenne wird als Komponente in das System integriert und das System hat als Schnittstelle dafür die Anforderungen an die Antenne. Jede der oben vorgestellten Antennen kann über diese Schnittstelle an das System angeschlossen werden. Des weiteren hat das System die Anforderungen an einen Verstärker als Schnittstelle. Diese werden in diesem Fall von einem Transmitter erfüllt.

Das System selbst bedient die Anforderungen der darüber liegenden Gesamtsystemebene. Es liefert eine Übertragungsrate für den Preis der Systemmasse und der verbrauchten Leistung. Des weiteren ist die Wellenlänge als externe Randbedingung vorgegeben. Durch die Synthese der Komponenten zu einem Kommunikationssystem entsteht die Funktionalität der Datenübertragung (`downlinkRate`) als Mehrwert im Gegensatz zu den Funktionen der einzelnen Komponenten. Durch die Synthese entsteht ein Gleichungssystem aus den analytischen Auslegungsgleichungen der Lösungsprinzipien.

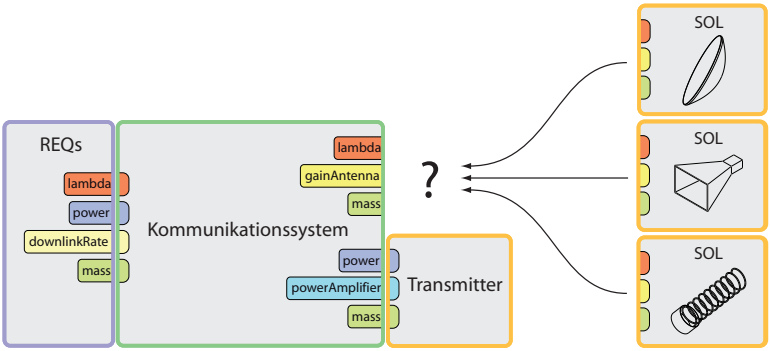


Abbildung 4.15: Integration des Lösungsprinzips auf Systemebene

Synthese Kommunikationssystem. Setzt man die Gleichungen der Lösungsprinzipien des Kommunikationssystems ineinander ein, so ergibt sich ein Lösungspfad zur Auslegung der Systemparameter. Für ein Kommunikationssystem mit einer Parabolantenne ist der Lösungspfad in Abb. 4.16 dargestellt.

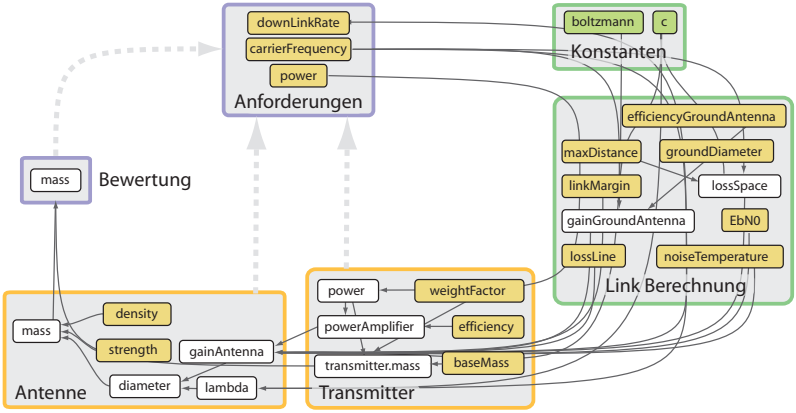


Abbildung 4.16: Lösungspfad des Kommunikationssystems als Entwurfszyklus

Als Eingabewerte in das System werden die Übertragungsrate, die Frequenz und der Energiebedarf ausgewählt. Aus diesen Vorgaben lässt sich zunächst die Strecke zur Bodenstation berechnen. Dann kann der Transmitter ausgelegt werden. Nach der Auslegung des Transmitters wird die Antenne ausgelegt und die Gesamtmasse des Systems bestimmt. Damit ist die Berechnung des Systems abgeschlossen.

Analyse des Systems. Für die Auswahl zwischen den verschiedenen Systemtopologien sind in Abb. 4.17 die Massen zweier Systemtopologien über der Datenrate und der vom System benötigten Leistung aufgetragen. Die Masse ist von oben (0kg) nach unten (10kg) aufgetragen. Die Achsen „Leistung“ und „Datenrate“ sind jeweils logarithmisch aufgetragen.

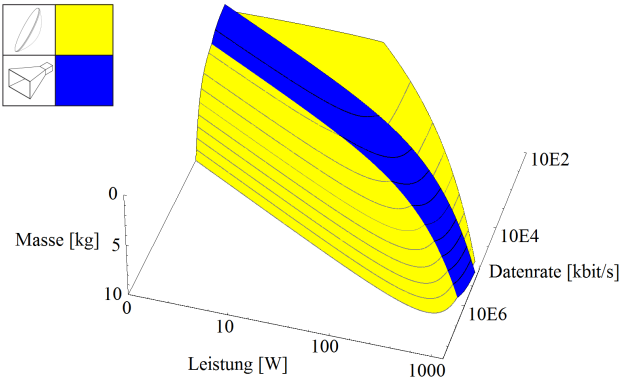


Abbildung 4.17: Zustandsflächen des Systems mit Horn- und Parabolantenne

Für die Systemtopologie mit einer *ParabolicAntenna* und einem *Transmitter* ergibt sich die in Abb. 4.17 gelb dargestellte Zustandsfläche. Jeder Punkt dieser Fläche repräsentiert ein Kommunikationssystem mit dieser Kombination an Lösungsprinzipien. Für die Systemtopologie mit einer *PyramidalHornAntenna* und einem *Transmitter* ergibt sich die in Abb. 4.17 blau dargestellte Zustandsfläche. Die Linien auf den Zustandsflächen sind Linien gleicher Masse im Abstand von 1 kg.

Für hohe Leistungen und niedrige Datenraten wird eine Begrenzungsfunktion eingeführt, die in der gelben Zustandsfläche sichtbar wird, da Systeme in diesem Bereich in der Realität nicht von Interesse sind. Die Auslegungsgleichungen für die Hornantenne ergeben erst ab einer Mindestgröße der Antenne reale Werte und daher hat die blaue Zustandsfläche eine Grenzlinie, ab welcher die Antenne nicht mehr ausgelegt werden kann. Die Parabolantenne wird hier bis auf sehr kleine Durchmesser skaliert, auch wenn in der Realität eine Mindestgröße notwendig ist. Dies spielt jedoch für die Form der Fläche keine große Rolle, da lediglich das Plateau mit den leichten Systemen etwas tiefer liegen würde.

In Abb. 4.17 kann man sehen, dass sich die beiden Zustandsflächen gegenseitig durchdringen. Die blaue Zustandsfläche liegt in einem schmalen Bereich über der gelben Zustandsfläche. Dies bedeutet, dass in diesem Bereich das System mit einer Hornantenne leichter ist als das System mit einer Parabolantenne. Nach hinten, in Richtung kleinerer Datenraten, ist die blaue Zustandsfläche zwar abgeschnitten, jedoch liegt die Hinterkante der blauen Zustandsfläche nach wie vor oberhalb der gelben Fläche. Dies bedeutet, dass auch in diesem Bereich das System mit der Hornantenne leichter ist als das System mit Parabolantenne. Nach vorne hin, in

Richtung größerer Datenraten, verschwindet die blaue Fläche unter der gelben. Dies bedeutet, dass in diesem Bereich das System mit der Parabolantenne leichter ist als das System mit der Hornantenne.

Die Linie, entlang der die blaue Zustandsfläche unter der gelben verschwindet, ist die Grenzlinie, an der ein Wechsel des leichtesten Systems stattfindet. In einer Entwurfssituation, in der das leichteste System ausgewählt werden soll, verläuft entlang dieser Grenzlinie die Entscheidung zwischen dem System mit der Parabolantenne und dem System mit der Hornantenne. In dem Bereich, in dem die blaue Zustandsfläche oberhalb der gelben liegt, fällt die Wahl auf eine Hornantenne. In dem Bereich, in dem die gelbe Zustandsfläche oberhalb von der blauen liegt, fällt die Wahl auf eine Parabolantenne.

In Abb. 4.18 ist die Masse des Kommunikationssystems mit einer Helixantenne dargestellt. Die Achsen „Leistung“ und „Datenrate“ sind im gleichen Wertebereich wie in den vorherigen Abbildungen logarithmisch aufgetragen. Die Zustandsfläche in Abb. 4.18 verläuft als schmales

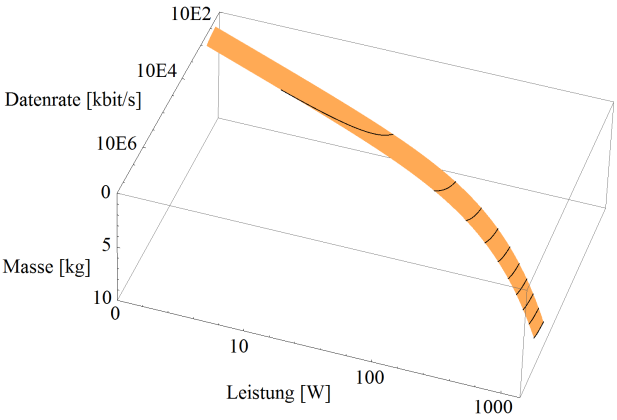


Abbildung 4.18: Zustandsfläche des Systems mit Helixantenne

Band über die verschiedenen Leistungsbereiche. Die Linien gleicher Masse sind nur als kurze Striche zu erkennen. Die Breite der Zustandsfläche ist in den Beschreibungsgleichungen der Helixantenne begründet. Die Gültigkeit der Gleichungen wird in [31] zwischen einer Windungszahl von 7 und 35 angegeben. Die Vorderkante der Zustandsfläche entspricht einem Kommunikationssystem mit einer Helixantenne mit 35 Windungen. Die Hinterkante entspricht einem Kommunikationssystem mit einer Helixantenne mit 7 Windungen.

Trägt man die gezeigten drei Zustandsflächen und die Zustandsflächen für die Systeme mit TWT in einer Grafik auf, so ergibt sich Abbildung 4.19. Die grüne Zustandsfläche im Bereich hoher Leistungen und hoher Datenraten repräsentiert die Masse des Kommunikationssystems mit Hornantenne und Wanderfeldröhre. Der orangefarbene Streifen in der grünen Zustandsfläche stellt ein Kommunikationssystem mit Helixantenne und Wanderfeldröhre dar. Man kann

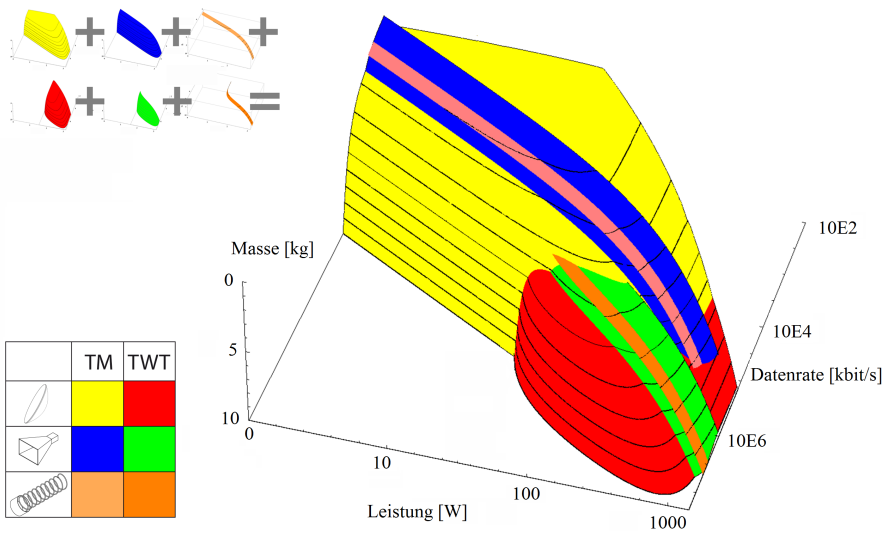


Abbildung 4.19: Zustandsflächen aller Systemtopologien

gut erkennen, wie sich die höhere Grundmasse der Wanderfeldröhre auf das System auswirkt. Alle Zustandsflächen dieser Art steigen nicht so hoch an wie die Zustandsflächen der Systeme mit nur einem Transmitter. Die höhere Effizienz der Wanderfeldröhre ermöglicht jedoch dafür Kommunikationssysteme mit 1 kW und mehr.

Auch in dieser Abbildung ist wie in Abbildung 4.17 zu erkennen, dass sich die einzelnen Zustandsflächen durchdringen. Die Zustandsflächen der Systeme mit Helixantenne tauchen jeweils in die Zustandsflächen der Systeme mit Hornantenne ab. Die Zustandsflächen der Systeme mit Wanderfeldröhre dringen in Richtung geringerer Leistungen in die Zustandsflächen der Systeme ohne Wanderfeldröhre ein. Um für einen bestimmten Auslegungspunkt die leichteste Systemtopologie herauszufinden, muss man nur die oberste Zustandsfläche bestimmen.

In Abb. 4.20 ist eine Ansicht auf Abb. 4.19 von der Richtung der geringsten Masse aus dargestellt. Dies bedeutet, dass in jedem Punkt nur die leichteste Systemkombination zu sehen ist. Die Abbildung gilt in einem Bereich der Systemmasse bis 10 kg. Die Zustandsflächen werden in dieser Abbildung erweitert, um die jeweils leichteste Systemtopologie in einem Bereich zu repräsentieren. Die blaue Zustandsfläche ist in Richtung niedrigerer Datenraten erweitert und deckt den gesamten hinteren Bereich ab. Diese Erweiterung resultiert aus der Tatsache, dass für die Darstellung der für die Systemtopologie-Auswahl entscheidenden Gebiete, die Zustandsflächen aus Abb. 4.19 nach hinten auf die Ebene der Achsen Masse und Leistung projiziert werden. Diese Projektion ergibt sich aus der Annahme, dass gegenüber einem System mit einer niedrigeren Datenrate ein System geringerer Masse und höherer Datenrate bevorzugt wird.

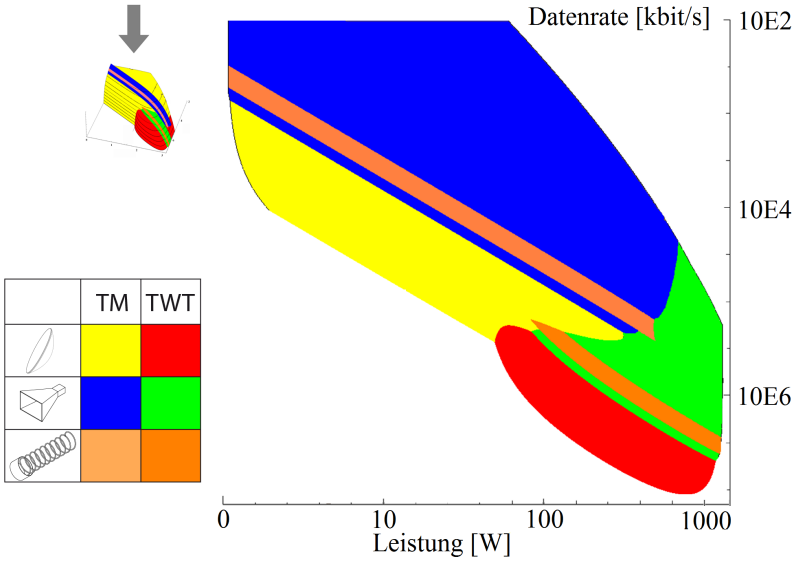


Abbildung 4.20: Zustandsflächen zur Systemtopologie-Auswahl

In der Abbildung ist sehr gut zu sehen, dass es Gebiete gibt, in denen eine Systemtopologie ohne naheliegende Nachbarn vorherrscht. Es gibt jedoch auch Gebiete, in denen häufige Wechsel zwischen den Systemtopologien stattfinden. Besonders hervorzuheben sind die Punkte, an welchen drei Systemtopologien aufeinander treffen. An diesen Punkten kann eine minimale Abweichung vom Auslegungspunkt zu einer anderen Systemtopologie-Auswahl führen. Diese Ansicht bietet eine generelle Entscheidungsmethode zwischen verschiedenen Systemtopologien in einem Auslegungspunkt. Diese Sichtweise auf den Entwurf bietet jedoch nur eine Analyse über drei Variablen aus einer größeren Menge an veränderlichen Größen. Um die Sensitivität aller Variablen einer Systemtopologie in einem Arbeitspunkt zu untersuchen, wird eine Sensitivitätsanalyse durchgeführt.

Sensitivitätsanalyse. Eine Sensitivitätsanalyse besteht in der lokalen Ableitung der Variablen des Entwurfs nach den Randbedingungen. Diese Methode hat den Vorteil, dass alle Variablen des Entwurfs auf einmal betrachtet werden können. Möglich ist dies durch eine von Bölling [9] entwickelte Darstellung in einer sogenannten Heatmap. In der Heatmap werden die Ableitungen dimensionslos normiert dargestellt. Die Normierung wird im Arbeitspunkt vorgenommen:

$$\text{Normierter Wert} = \frac{x_0}{y_0}\left(\frac{\partial y}{\partial x}\right)_0 \tag{4.25}$$

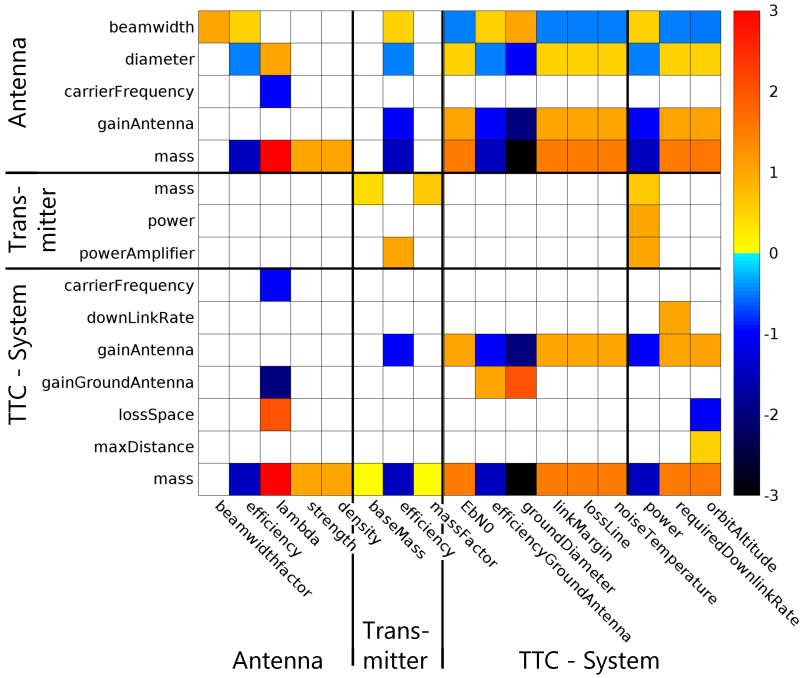


Abbildung 4.21: Heatmap des Kommunikationssystems mit Parabolantenne nach [9]

In Abbildung 4.21 ist eine Heatmap für das Kommunikationssystem mit Parabolantenne und Transmitter dargestellt. Entlang der horizontalen Achse sind die Eingangswerte x für den Entwurf aufgetragen. Entlang der vertikalen Achse sind die berechneten Variablen y aufgetragen. Wenn eine Kopplung zwischen den beiden Variablen vorliegt, es also möglich ist $\partial y/\partial x$ zu bilden, so ist das Ergebnis der Ableitung im Arbeitspunkt farbig dargestellt. Die Farbpalette reicht von -3 bis 3. Sie geht dabei von weiß (0) bis gelb (10^{-8}), von gelb bis orange (1) und von orange bis hin zu rot (3). Im negativen Bereich geht es von weiß über türkis und blau bis hin zu schwarz in denselben Intervallen. Die Werte der Farben sind normiert auf den Arbeitspunkt und prozentual angegeben. Die Werte gelten jeweils unter sonst gleichen Bedingungen. Eine 1 in der Heatmap entspricht einer Änderung des Zielwertes um 10% bei einer Änderung des Ausgangswertes um 10%. Eine 2 besagt, dass sich der Zielwert im gleichen Intervall um 20% ändert.

Betrachtet man die Zeile der Antennenmasse `mass` oberhalb der schwarzen Trennlinie, so sieht man im zweiten Feld eine blaue Einfärbung. Die blaue Einfärbung steht für einen Wert von -1 und bedeutet, dass eine Steigerung der Antenneneffizienz um 10% die Masse der Antenne um -10% beeinflusst. Dies entspricht durchaus unserer Erwartung. Das rote Feld daneben sagt aus,

dass die Masse der Antenne bei einer Steigerung der Wellenlänge um 10% um 30% ansteigt. Auch dies überrascht nicht, da der Durchmesser der Antenne, wie in Gleichung (4.4) zu sehen, mit der Wellenlänge ansteigt.

Betrachtet man die Matrix Spaltenweise, so stellen zwei Spalten, deren Farbwerte in jeder Zeile das gleiche Vorzeichen haben, gleichgesinnte Entwurfsgrößen dar. Spaltenpaare, deren Farbwerte in jeder Zeile ungleiche Vorzeichen haben, stellen hingegen Gegenspieler im Entwurf dar [9]. In Abb. 4.21 ist zum Beispiel die Leistung der Systems `power` ein Gegenspieler zur Rauschtemperatur `noiseTemperature`. Die Rauschtemperatur und der Verlust in den Leitungen `lossLine` hingegen sind Gleichgesinnte. Dies kann man auch gut in Gleichung (4.1) überprüfen, da dort die Rauschtemperatur und der Leitungsverlust (als Wert größer 1) beide im Nenner stehen, die Leistung (in diesem Fall des Verstärkers) jedoch im Zähler.

4.2 Beschreibung der Subsysteme

Im letzten Abschnitt wurde mit einer Entwurfssprache ein Subsystem für einen Satelliten aus verschiedenen Komponenten aufgebaut. Dabei wurden auch die einzelnen Komponenten ausgelegt. In diesem Abschnitt wird nun die methodische Grundlage für den Zusammenbau eines Satelliten aus verschiedenen Subsystemen in verschiedenen Entwurfssprachen erklärt.

Um die Entwurfssprachen für Subsysteme austauschbar zu machen, gibt es zwischen ihnen keine Abhängigkeiten. Dadurch ist es möglich, verschiedene Satellitenprojekte mit verschiedenen Kombinationen von Subsystem-Entwurfssprachen zu realisieren. In Abb. 4.22 ist die dafür verwendete Hierarchie der verschiedenen Entwurfssprachen dargestellt.

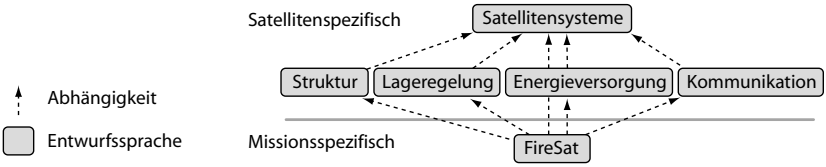


Abbildung 4.22: Hierarchie der Satelliten-Entwurfssprachen

Die oberste Entwurfssprache ist die Systemsprache. Anhand der Systemsprache werden zunächst die Konzepte für die Integration der Subsystem-Entwurfssprachen vorgestellt. Anschließend wird die Entwurfssprache für Struktur mit Ihren Klassen erläutert. Die Entwurfssprache für das Kommunikationssystem wurde im vorangegangenen Abschnitt behandelt. Das Lageregelungssystem wird ausführlicher besprochen, da es die Grundlagen für ein Simulationsmodell enthält. Als viertes Subsystem wird die Energieversorgung behandelt. Im letzten Unterabschnitt wird dann die missionsspezifische FireSat-Entwurfssprache dargestellt. In dieser Entwurfssprache werden alle darüber liegenden Sprachen integriert und der Satellit aufgebaut.

4.2.1 Satellitensysteme

Die Satellitensystemsprache (oben in Abb. 4.22) ist die gemeinsame Obermenge aller Subsystem-Entwurfssprachen. Um die verschiedenen Subsysteme in einen Satellit integrieren zu können, muss deren Auslegung aufeinander abgestimmt werden. Die verschiedenen Bedarfe der Subsysteme an Energie, Ausrichtung, Befestigung oder Kommunikation und Steuerung müssen gesammelt und dann an das jeweils zuständige Subsystem weitergegeben werden.

Dafür wird zum Beispiel der Energiebedarf aller Verbraucher aggregiert und dann an das Energieversorgungssystem weitergegeben. Die Systembudgets sind immer nach dem gleichen Muster aufgebaut. Jedes Budget besteht aus einer *Element*-Klasse, einer *Aggregation*-Klasse und einer *Budget*-Klasse. In Abb. 4.23 sind die Klassen des mechanischen Budgets abgebildet.

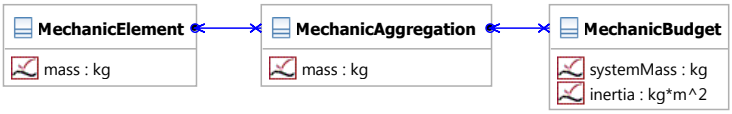


Abbildung 4.23: Mechanisches Budget

Die *MechanicElement*-Klasse wird auf jedes Element im Entwurf vererbt, das die entsprechende Eigenschaft besitzt. Im Fall des Massenbudgets in Abb. 4.23 ist dies jedes massebehaftete Bauteil. Neben der *MechanicElement*-Klasse gibt es eine *Aggregation*-Klasse. Diese sammelt die Daten jedes Elements und bildet entsprechend der jeweiligen Aggregationsvorschrift einen Gesamtwert. Bei den hier gezeigten Budgets ist der Gesamtwert jeweils die Summe aller Teile, es gibt jedoch auch Fälle, in denen der Zielwert das kleinste aller Teile ist (z.B. bei den Ausrichtungsanforderungen des sog. „Pointing Budget“) oder Budgets wie z.B. die Trägheitsmomente, in denen der Steinersche Anteil bei der Summation berücksichtigt werden muss.

In Abb. 4.24 ist das Energie-Budget dargestellt. Jedes *PowerElement* kann verschiedene

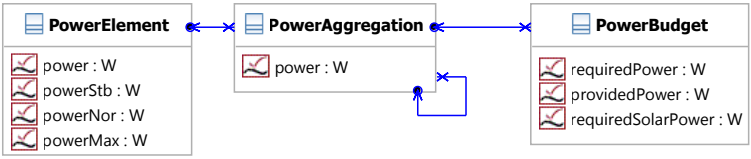


Abbildung 4.24: Energie-Budget

Verbrauchswerte angeben. Die Werte `powerStb`, `powerNor` und `powerMax` dienen dazu, den Leistungsbedarf der einzelnen Geräte in den verschiedenen Betriebsmodi des Satelliten zu unterscheiden. So kann für jedes Gerät ein Standby-Verbrauch, ein Normverbrauch und ein Verbrauch unter Spitzenlast angegeben werden.

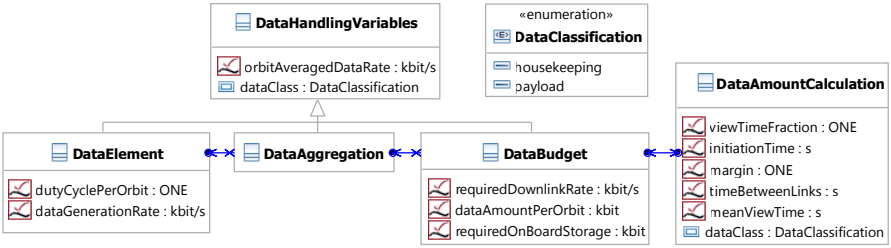


Abbildung 4.25: Datenverarbeitungs-Budget

In Abbildung 4.25 sind die Klassen des Datenverarbeitungs-Budgets dargestellt. In der Oberklasse *DataHandlingVariables* sind die Variablen für das Budget definiert. Die Klasse *DataElement* wird auf jedes Element vererbt, das entweder Housekeeping- oder Nutzlastdaten produziert. Die Ergebnisse werden von der Entwurfssprache zur Datenverarbeitung verwendet, um die bordeigene Datenspeicherung und Verarbeitung auszulegen.

In Abbildung 4.26 ist die Verwendung der Budgets am Beispiel des Kommunikationssystems dargestellt. Die Klasse *SatelliteLink* erbt von den Aggregationsklassen der verschiedenen

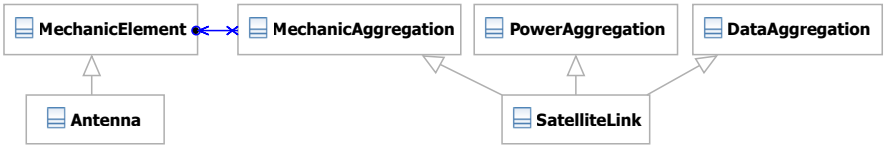


Abbildung 4.26: Beispielhafte Anwendung der Budgets

Budgets. Die Klasse *Antenna*, die Oberklasse aller Antennen dieser Entwurfssprache, erbt von *MechanicElement*. Wird nun die Instanz der Antenne über die abgebildete Assoziation mit der Instanz von *SatelliteLink* verbunden, so wird die Masse der Antenne automatisch zu der Masse des Subsystems hinzugezählt. Auf diese Weise können die verschiedenen Instanzen mehrere Rollen gegenüber einer anderen Instanz haben und daher auch über mehrere Links miteinander verbunden sein.

4.2.2 Struktur

Die Entwurfssprache für Struktur kann eine quaderförmige Bus-Struktur in Abhängigkeit der Randbedingungen auslegen. Für die Auslegung der Struktur werden keine Festigkeitsanforderungen berücksichtigt, da das für die Bewertung notwendige (FE-) Modell nicht Teil dieses Beispiels ist. Durch den modularen Aufbau der Entwurfssprachen könnte die Struktur jedoch jederzeit durch andere Varianten ausgetauscht werden. In Abb. 4.27 ist das Klassendiagramm

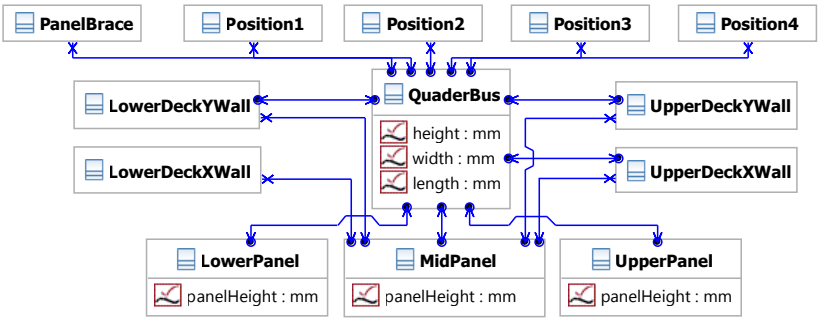


Abbildung 4.27: Klassen der Struktur

der Entwurfssprache dargestellt. Die Klasse *QuaderBus* gibt mit den Eigenschaften Länge, Höhe und Breite die groben Abmaße des Satelliten vor. Der Satellit besteht dabei im Wesentlichen aus drei Ebenen: Das unterste *LowerPanel*, das mittlere *MidPanel* und oben liegend das *UpperPanel*. An den vier Ecken der Ebenen sind Streben (*PanelBrace*) über die *Position1* bis *Position4* platziert. In der unteren und der mittleren Ebene gibt es die Möglichkeit zwei Wände (*LowerDeckXWall* und *LowerDeckYWall* bzw. *MidDeckXWall* und *MidDeckYWall*) einzubauen.

In Abbildung 4.28 ist der Ablauf zum Aufbau der Struktur abgebildet. Als erstes wird der

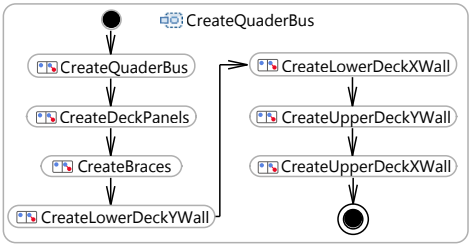


Abbildung 4.28: Aufbausequenz der Struktur

QuaderBus instanziiert. Dann werden die verschiedenen Decks und die Streben eingebaut und zum Schluss werden auf der oberen und der unteren Ebene senkrechte Wände eingebaut.

Durch diese inkrementelle Vorgehensweise wird erreicht, dass wenn man bestimmte Teile nicht einbauen möchte, diese durch Auslassen der Regel einfach ausgeschaltet werden können. Die Wände und Streben werden über Gleichungen skaliert und in der Größe auf den aktuellen Entwurfspunkt angepasst. In Abb. 4.29 sind zwei Varianten der Struktur unter verschiedenen Randbedingungen dargestellt. Die Struktur ist über Länge, Höhe und Breite skalierbar. Auch die Abstände zwischen den Decks sind über die Parameter einstellbar. An verschiedenen Stellen

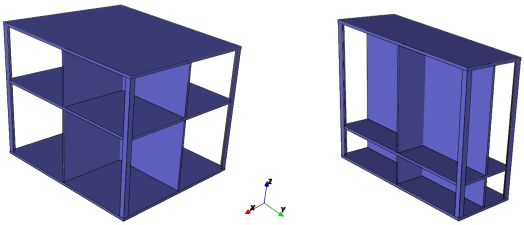


Abbildung 4.29: Varianten der Struktur

können über Regeln Durchgänge für Kabeln oder Leitungen eingebracht werden. Die Struktur bietet damit eine Aufnahmemöglichkeit für alle Komponenten einer Systemtopologie.

4.2.3 Lageregelung

Das Lageregelungssystem des Satelliten ist für die Bestimmung und Kontrolle der Lage des Satelliten im Raum zuständig. Für die Lagebestimmung gibt es verschiedene Sensoren mit unterschiedlichen Messprinzipien. Diese Sensoren senden ihre Informationen an den Bordcomputer, der daraus die Lage des Satelliten berechnet. Stimmt die Lage des Satelliten nicht mit der gewünschten Lage überein, so werden vom Bordcomputer die Aktuatoren der Lageregelung angesteuert, um die Ausrichtung zu ändern.

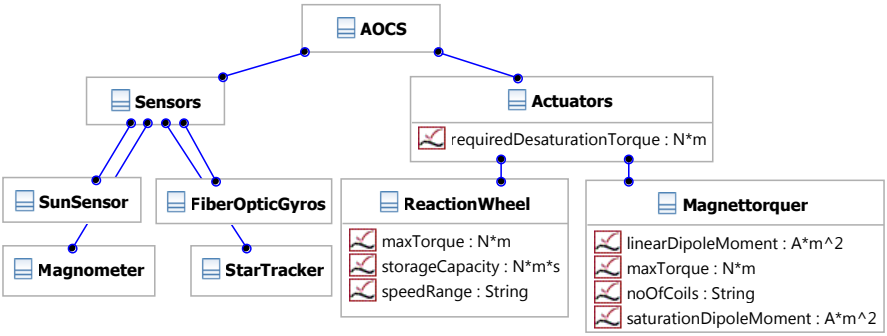


Abbildung 4.30: Klassen des Lageregelungssystems

In der Modellierung des Lageregelungssystems sind verschiedene Sensoren und Aktuatoren berücksichtigt. Das System kann aus Kombinationen dieser Bauteile aufgebaut werden. In Abb. 4.30 sind einige Klassen des Lageregelungssystems dargestellt. Die oberste Klasse *AOCS* repräsentiert das Lageregelungssystem (Attitude and Orbit Control System). Unterhalb der Systemklasse teilt sich die Hierarchie auf in Aktuatoren (*Actuators*) und Sensoren

(*Sensors*). Die Sensoren bestehen aus dem Sonnensensor (*SunSensor*), dem Magnetfeldsensor (*Magnetometer*) und den Sternkameras (*StarTracker*) als Lagesensoren und dem faseroptischen Kreisel (*FiberOpticGyro*) als Drehratensensor. Als Aktuatoren sind in diesem Beispiel Reaktionsräder (*ReactionWheels*) als Momentenspeicher und Magnettorquer (*Magnettorquer*) als Momentenerzeuger eingebaut.

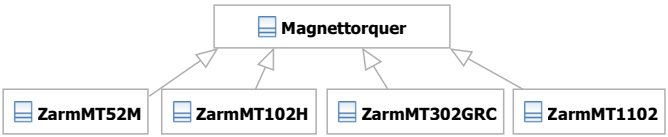


Abbildung 4.31: Verschiedene Magnettorquer-Klassen

In Abb. 4.31 ist die Magnettorquerklasse mit vier speziellen Bauteilen als Unterklassen dargestellt. Die hier verwendeten Magnettorquer sind kommerziell erhältliche Komponenten nach [74]. Die Klassen der Komponenten erben von der Oberklasse die Eigenschaften eines Magnettorquers wie zum Beispiel das lineare Dipolmoment (`linearDipoleMoment`) oder die Anzahl der Spulen (`noOfCoils`). Sie unterscheiden sich lediglich in den Werten, welche diesen Eigenschaften zugewiesen werden.

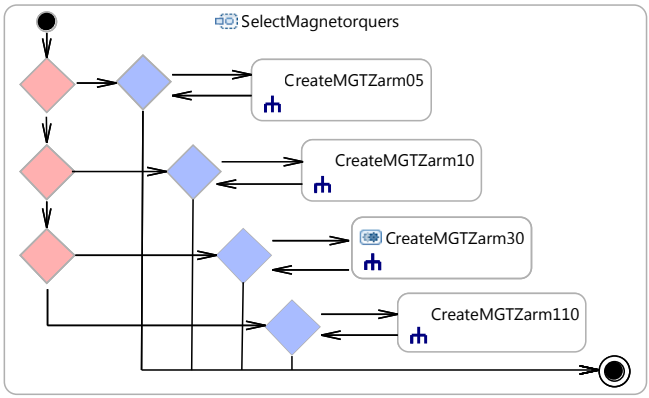


Abbildung 4.32: Aktivtät zum Einbau des Magnettorquers

Die Auswahl zwischen diesen Komponenten erfolgt aufgrund einer Entscheidungslogik, die in Abb. 4.32 dargestellt ist. Die Aktivität beginnt oben links und durchläuft die roten Entscheidungsknoten[3]. In diesen Knoten wird das maximale Moment (`maxTorque`) des

[3]Diese „Decision Nodes“ der UML werden in Entwurfssprachen mit OCL-Constraints versehen um Bedingungen in Abhängigkeit des aktuellen Entwurfsstandes angeben zu können.

speziellen *Magnettorquer* mit der Anforderung `requiredDesaturationTorque` der *Actuators* verglichen. Genügt das Moment des Torquers, so folgt der Fluss dem Pfeil nach rechts. Im blauen Entscheidungsknoten gibt es dann einen Zähler, der das Unterprogramm „CreateMGTZarm..“ so oft aufruft, bis die geforderte Anzahl an Magnettorquern eingebaut ist.

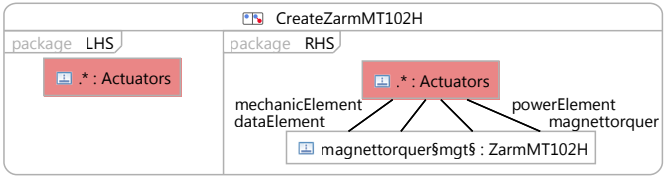


Abbildung 4.33: Regel zum Einbau des Magnettorquers

Der Einbau eines Magnettorquers ist in Abb. 4.33 dargestellt. Die vier Links zwischen den beiden Instanzen legen die Rollen fest, welche der Magnettorquer für das Aktuator-System spielt. Der Magnettorquer ist zunächst ein `magnettorquer` für den Aktuator. Des weiteren ist er ein `mechanicElement`, dies bedeutet, dass die Masse des Magnettorquers zur Masse des Aktuator-Systems beiträgt. Der Link `powerElement` propagiert den Leistungsbedarf an das *PowerBudget* und der Link `dataElement` verknüpft ihn mit dem *DatenBudget*. So werden in dieser Einbauregel die verschiedenen Rollen der Komponenten gegenüber dem System festgelegt.

Neben der funktionalen und der geometrischen Repräsentation wird für die Komponenten des Lageregelungssystems ein Aktivitätsdiagramm für die Abbildung des Systemverhaltens aufgebaut. Das Modell wurde aufbauend auf einem Beispielmodell in Matlab Simulink [43] in einer begleitenden Diplomarbeit [53] erweitert und in die Entwurfssprache eingepflegt. In Abb. 4.34 ist die oberste Ebene des Aktivitätsdiagramms vereinfacht dargestellt.

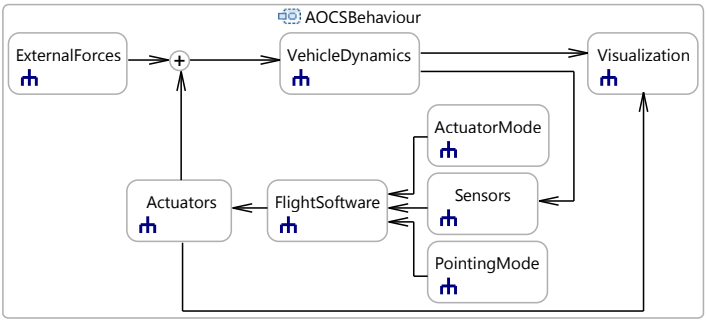


Abbildung 4.34: Aktivität zur Simulation der Lageregelung

Die Bewegung des Satelliten wird mit einer Starrkörperbewegung in der Aktivität *Vehicle-Dynamics* abgebildet. Die Dreh- und Lagewerte des Satelliten werden an die Aktivität *Sensors* weitergegeben und dort durch die Modelle der Sensoren in Messwerte umgewandelt. Diese werden an die Flugsoftware zur Berechnung der Aktuatorkommandos weitergegeben. In der Flugsoftware, der Regelung des Satelliten, wird über ein einfaches PD-Reglerkonzept die Ansteuerung der Aktuatoren berechnet. Die Steuersignale werden an die Aktuatoren weitergegeben und von den Modellen der Aktuatoren werden die resultierenden Kräfte berechnet. Danach werden die externen Randbedingungen aufgeprägt und der Kreislauf beginnt von neuem.

Diese Aktivität und die verschiedenen Subaktivitäten können nach Simulink exportiert und dort simuliert werden. In [53] sind noch weitere Aktuatoren implementiert. Beispielhaft wird in Abb. 4.35 der Einbau des *Magnettorquers* in die Subaktivität *Actuators* aus Abb. 4.34 dargestellt.

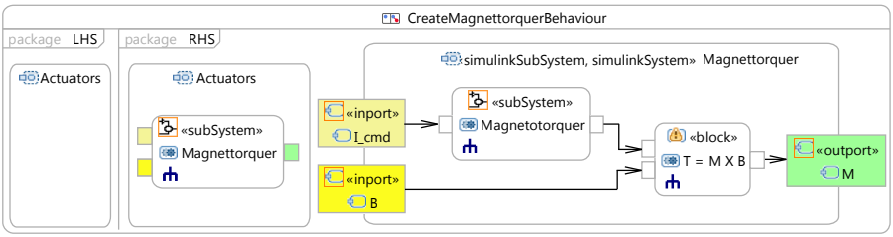


Abbildung 4.35: Regel zum Einbau des Magnettorquers in die Simulation

Auf der linken Seite der Regel wird zunächst die Subaktivität *Actuators* aus Abb. 4.34 gesucht. Auf der rechten Seite wird dann in dieser Subaktivität eine „CallBehaviourAction“ mit dem Namen Magnettorquer eingebaut. Diese CallBehaviourAction ruft eine weitere Subaktivität auf, welche auf der rechten Seite in der Regel dargestellt ist. Die In-Ports sind dort aufgelöst und die innere Struktur der Aktivität ist ebenfalls sichtbar. Das Signal wird ausgehend von dem Kommando „I_cmd“ in einem „ForkNode“ auf zwei Stränge verteilt. Im oberen Strang wird das resultierende magnetische Moment in Abhängigkeit des Eingangsstroms berechnet. Dieses Moment wird dann im „Outport“ „M_MT“ ausgegeben und im hinteren Block mit der magnetischen Flussdichte der Erde zu einem Moment auf den Satelliten umgerechnet. Dieses Moment wirkt sich dann auf die Bewegung des Satelliten aus. Im unteren Strang wird zusätzlich dazu noch die benötigte Leistung des Magnettorquers berechnet.

In der in Abb. 4.36 dargestellten Regel wird das Verhalten des Magnettorquers innerhalb der *Actuators*-Aktivität in den Ablauf eingebaut. Der Block „ActBS“ ist ein Verteiler für die Signale, die vom Gesamtsystem an die Aktuatoren übergeben werden. Die speziellen Kommandos des Magnettorquers können über die beiden gelb eingefärbten Ports identifiziert werden. Nach der Berechnung des Moments in der in Abb. 4.35 dargestellten Aktivität, wird das Ergebnis über den Block „Torquessum“ mit den Momenten der anderen Aktuatoren aufsummiert und an die

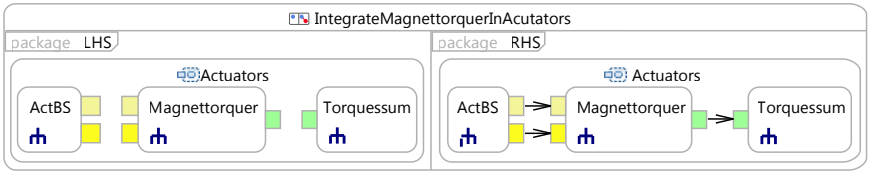


Abbildung 4.36: Regel zur Integration des Magnettorquers in Simulationsablauf

Systemsimulation übergeben.

Das vorgestellte Vorgehen ermöglicht es, nicht nur die geometrischen und funktionalen Aspekte des Systems in der Entwurfssprache abzubilden, sondern auch das Verhalten konsistent im gleichen Modell mit aufzubauen. So können verschiedene Varianten des Subsystems in einen Satellit eingebaut und das resultierende Systemverhalten gleich überprüft werden.

4.2.4 Energieversorgung

Das Energieversorgungssystem ist für die Produktion, Speicherung und Verteilung elektrischer Energie zuständig. In Abb. 4.37 sind die zentralen Klassen des Systems dargestellt. Die Systemklasse *PowerSystem* holt sich vom *PowerBudget* den Energiebedarf des Gesamtsystems und berechnet daraus den Bedarf während der Tagesphase `powerDaylight` sowie während der Schattenphase `powerEclipse`. Aus diesen beiden Anforderungen lässt sich über die Angabe der Verluste auf den beiden Pfaden (in der Tagesphase direkt, in der Schattenphase mit Laden und Entladen der Batterie) der Leistungsbedarf des *PhotovoltaicSystem* bestimmen. Für die Stromversorgung in der Schattenphase ist das *BatterySystem* zuständig. Das *BatterySystem*

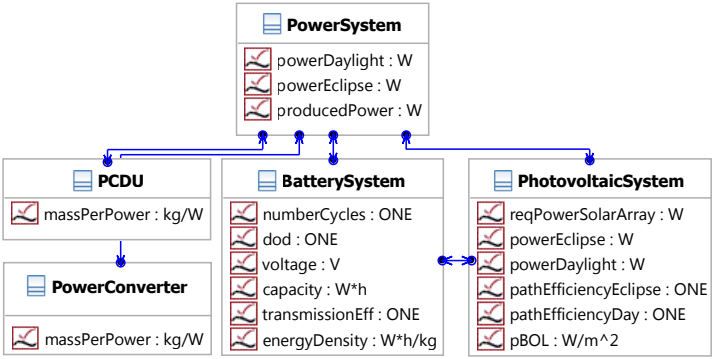


Abbildung 4.37: Klassen der Energieversorgung

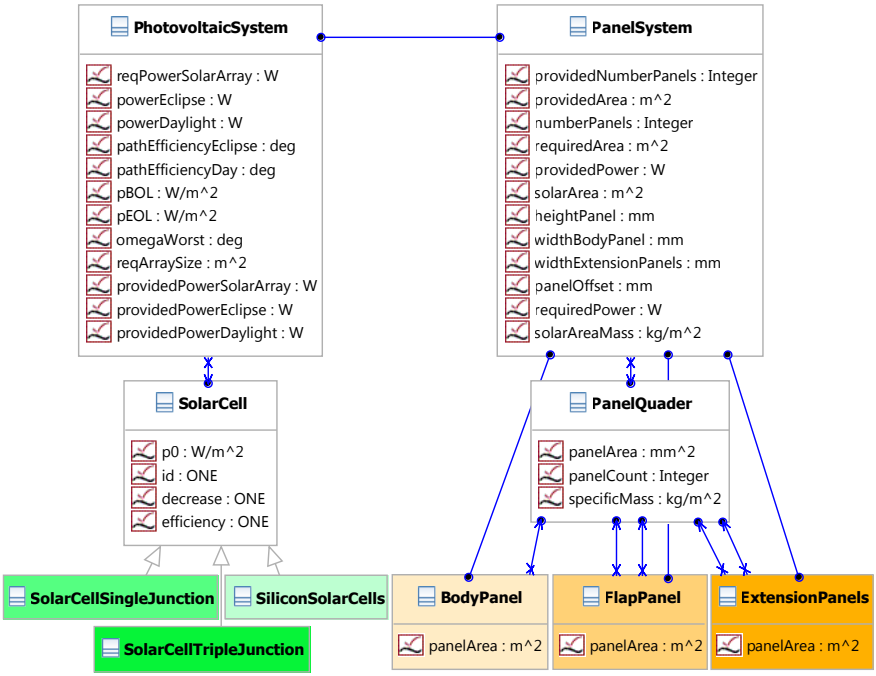


Abbildung 4.38: Klassen des Photovoltaik-Systems

berechnet aus der Missionsdauer die Anzahl der Ladezyklen (`numberCycles`). Aus der Haltbarkeit der eingesetzten Batterie ergibt sich damit der Faktor `dod` - „depth of discharge“, also die Entladungstiefe der Batterie. Die „Power Control and Distribution Unit“ - *PCDU* geht in dieser Modellierung über ein Leistungsgewicht (`massPerPower`) in die Masse des Gesamtsystems ein. Der Faktor `massPerPower` wird durch eine Interpolation von bestehenden Systemen bestimmt. Ebenso wird die Masse des *PowerConverter*, der die Spannungssteuerung der Solarzellen vornimmt, über ein Leistungsgewicht (`massPerPower`) mit der verarbeiteten Leistung bestimmt.

Das Photovoltaik-System ist in Abb. 4.38 genauer dargestellt. Es gibt eine Klasse *SolarCell*, in der das Attribut für die Effizienz (`efficiency`) der Solarzellen definiert ist. Von dieser Klasse erben drei verschiedene Typen von Solarzellen, eine Zelle auf Silikonbasis (*SiliconSolarCells*), eine Einschichtzelle auf GaAs-Basis (*SolarCellSingleJunction*) und eine Dreischichtzelle (*SolarCellTripleJunction*). Die drei Zellen haben jeweils unterschiedliche Werte für die Verfallszahl pro Jahr (`decrease`) und die Anfangseffizienz (`efficiency`).

Aus diesen Werten wird dann die spezifische Leistung zu Beginn, `pBOL` (Power Begin Of

Life) und zu Ende `pEOL` (Power End Of Life) der Mission berechnet. Aus der spezifischen Leistung am Ende der Lebensdauer (`pEOL`) wird dann die benötigte Fläche für die Solarzellen berechnet und an das *PanelSystem* weitergegeben.

Die Aufbaulogik der Solarpanele ist in Abb. 4.39 als Aktivitätsdiagramm dargestellt. Als

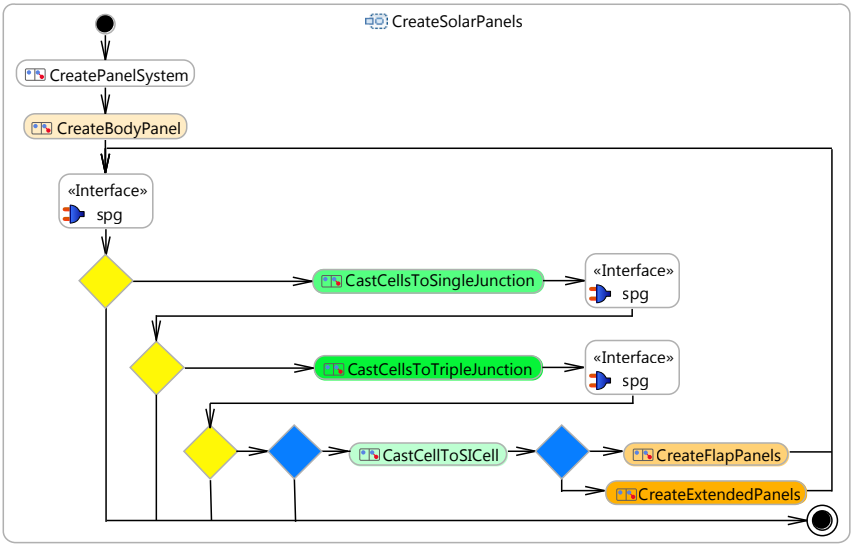


Abbildung 4.39: Aktivität zum Aufbau der Solarpanele

erstes wird das *PanelSystem* gebaut. Dann wird ein *BodyPanel*, ein fest auf der Rückseite des Satelliten montiertes Panel eingebaut. Nachdem dieses Panel eingebaut ist, werden die Gleichungen des Entwurfs mit einem Interfaceaufruf „spg“ gelöst. Danach wird die produzierte Leistung mit der benötigten Leistung verglichen.

Ist die produzierte Leistung größer als die benötigte, ist der Entwurf fertig und der Prozess folgt dem Pfeil zum Ende des Diagramms unten rechts. Reicht die produzierte Leistung nicht aus, werden nun zunächst die Solarzellen verbessert, indem die Instanz der Silikon-Solarzellen auf die Klasse der GaAs-Zellen geändert (gecastet) wird. Nun werden wieder die Gleichungen gelöst und überprüft, ob die Leistung ausreicht. Ist dies nicht der Fall, so werden die besten verfügbaren Solarzellen ausprobiert. Wenn auch diese nicht ausreichen, so werden zunächst wieder die einfachsten und damit auch günstigsten Zellen eingebaut. Dann werden die *FlapPanels*, ein klappbares Panel jeweils rechts und links vom *BodyPanel* eingebaut.

Die Logik entspricht dem Gedanken, dass es günstiger ist, weniger Panele und dafür teurere Zellen zu verwenden. Muss man mehr Panele einbauen, so kann man es zunächst wieder mit den günstigen Zellen versuchen.

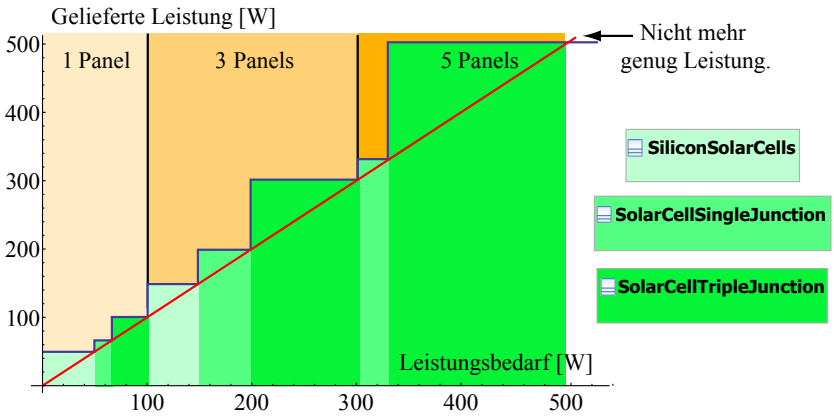


Abbildung 4.40: Leistung des Energieversorgungssystems über Leistungsbedarf

Aus dieser Logik resultiert das in Abb. 4.40 aufgetragene Verhalten des Energieversorgungssystems. In dieser Abbildung ist die gelieferte Leistung des Systems über dem Leistungsbedarf aufgetragen. Man sieht nun, dass die gelieferte Leistung treppenförmig ansteigt, da die Änderungsmöglichkeiten des Systems immer diskret sind. So sind die ersten beiden Stufen immer der Verbesserung durch bessere Zellen geschuldet und der nächste Schritt entsteht durch den Wechsel auf mehr Panele. Ab einem bestimmten Leistungsbedarf kann die Anforderung nicht mehr erfüllt werden, da die Anzahl der klappbaren Panele auf zwei auf jeder Seite beschränkt ist. Über den Entwurf eines Energieversorgungssystems kann man schließen, dass er umso günstiger ist, je näher die gebotene Leistung an der geforderten liegt, da ansonsten überschüssige Energie produziert wird.

4.2.5 FireSat Nutzlast

Die Nutzlast des Satelliten besteht aus einem Teleskop und der zugehörigen Elektronik für dieses Teleskop. Die Auslegung des Teleskops ist in [70, Kapitel 9.5] ausführlich beschrieben. In [70, Tabelle 9.15] werden die Grundgleichungen für die Berechnung des Teleskops beschrieben. Diese Gleichungen werden in die Entwurfssprache übernommen und so kann das Teleskop für jeden Entwurf unter neuen Randbedingungen ausgelegt werden. Die Berechnung wird dabei in drei eng zusammenhängende Klassen unterteilt, wie in Abb. 4.41 dargestellt. Die Klasse *OpticalInstrument* beinhaltet die geometrischen Parameter von der Schwadbreite `swathWidth` über die verschiedenen Blickwinkel des Teleskops `lookAngle` und `incidenceAngle` hin zur Größe eines Bodenpixels. Ein Bodenpixel ist das Rechteck auf der Erdoberfläche, welches nachher auf einen Pixel des Instruments projeziert wird. Die Seitenlänge dieses Rechtecks wird durch die Variablen `xPixel` und `yPixel` beschrieben. In der Klasse *SensorOptics* werden die

Abbildung 4.41: Nutzlast-Klassen

geometrischen Daten des Teleskops wie die Fokallänge `focalLength` und der Aperturdurchmesser `apertureDiameter` berechnet und die Wellenlänge `operatingWavelength` für das Instrument festgelegt.

In der Klasse *SensorRadiometry* werden die physikalischen Gegebenheiten des Instruments berechnet. Diese Klasse wird im Entwurf zweimal instanziiert. Die beiden Instanzen berechnen die Anzahl der Elektronen (`numElectrons`) im Sensor mit einer um 1 Kelvin unterschiedlichen Erdtemperatur als Basis. Aufgrund dessen kann dann die „Noise Equivalent Temperature Difference“ (`noiseEqTempDiff`), das sogenannte Rauschtemperaturäquivalent berechnet werden. Dies ist der Temperaturunterschied eines Bodenpixels, welcher für die Kamera gerade über der Rauschschwelle liegt und somit gemessen werden kann. Dieser Wert entspricht der Temperaturauflösung der Kamera und ist somit eine wichtige Auslegungsgröße für die Mission. Der Energiebedarf der Nutzlast wird in der Klasse *FireSatPayload* über eine Referenzmission nach [70, Seite 296] berechnet. Der Referenzwert wird über den Durchmesser des Teleskops festgelegt und dient auch zu einer ersten Massenabschätzung.

4.3 Enumeration mit Entwurfssprachen

In diesem Abschnitt wird an dem umfangreichen FireSat-Beispiel gezeigt, wie sich durch drei wesentliche Vorteile der Entwurfssprachen der Entwurfsprozess verbessern lässt. Nachdem die Modellierung der Subsysteme vorgestellt ist, folgt der Aufbau des FireSat-Gesamtsystems aus den gezeigten Komponenten. Die Enumeration beginnt mit der Synthese des Gesamtsystems.

Als erste Stärke wird gezeigt, wie durch die Mittel einer Entwurfssprache eine integrierte Prozesskette für die vollständige Vorauslegung eines Satelliten entsteht. In einer Entwurfssprache kann der Aufbau des Satelliten zentral gesteuert werden und somit die Entscheidungen zwischen den verschiedenen Lösungsprinzipien und Komponenten flexibel gewählt werden.

Auf die Synthese des Gesamtsystems folgt die Analyse einer Systemtopologie. Um den Entwurf des FireSat besser zu verstehen, wird zunächst eine Analyse der Nutzlast über einen breiten Auslegungsbereich durchgeführt. Danach wird eine Systemtopologie des Gesamtsystems analysiert. Als zweite wesentliche Stärke der Entwurfssprachen wird dabei die Möglichkeit erläutert, ein Gleichungssystem für den Gesamtentwurf automatisch zu erstellen. Durch dieses Gleichungssystems ist es möglich, die Masse des Gesamtsystems in Abhängigkeit der Anforderungen an den Satellit analytisch zu bestimmen und als Zustandsfläche darzustellen.

Die dritte Stärke einer Entwurfssprache, die in diesem Abschnitt demonstriert wird, ist die Abbildung des Entwurfsraums mittels Regeln. Damit können die Gleichungssysteme außerhalb ihres Gültigkeitsbereichs automatisch ausgetauscht werden. Aus diesem regelbasierten Wechsel zwischen verschiedenen Gleichungssystemen entsteht die Möglichkeit, den Entwurfsraum verschiedener Systemtopologien zu betrachten. In Abschnitt 4.3.4 wird anhand einer solchen Untersuchung gezeigt, wie Muster im Entwurfsraum ausfindig gemacht werden können. Im Folgenden wird zunächst die Synthese der Gleichungen erläutert.

4.3.1 Synthese Gesamtsystem

Der Aufbau des Gesamtsystems wird in einem Aktivitätsdiagramm der FireSat-Entwurfssprache gesteuert. In Abb. 4.42 ist die oberste Ebene dieses Aktivitätsdiagramms abgebildet. In dem Aktivitätsdiagramm werden ausschließlich Subaktivitäten der verschiedenen Entwurfssprachen aufgerufen. In der obersten Reihe werden zwei Aktivitäten der Systemsprache aufgerufen. Die erste Aktivität „CreateGlobals“ erstellt eine Instanz mit den verwendeten Naturkonstanten

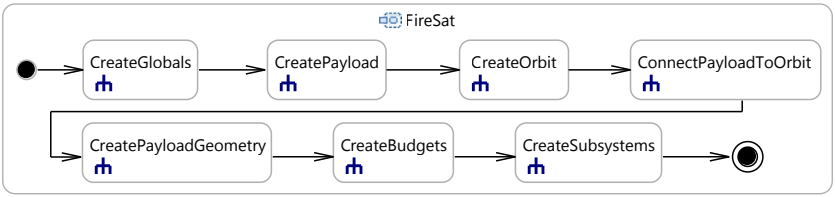


Abbildung 4.42: FireSat Aktivitätsdiagramm

und Variablen, auf die vom Gleichungssystem global zugegriffen werden kann. In der zweiten Aktivität „CreatePayload“ wird die Nutzlast aufgebaut. Danach wird eine Aktivität „CreateOrbit“ der System-Entwurfssprache aufgerufen und es werden verschiedene Instanzen erstellt, die den Orbit beschreiben. Die Instanzen der Nutzlast werden in „ConnectPayloadToOrbit“ mit den verschiedenen Orbit-Instanzen verbunden.

In der zweiten Reihe wird als erstes die Nutzlastgeometrie erstellt. Dann werden die verschiedenen Systembudgets aus der Systemsprache erstellt und mit der Nutzlast verbunden. Danach werden die Subsystem-Entwurfssprachen aufgerufen. Diese Subaktivität ist in Abb. 4.43 gesondert dargestellt.

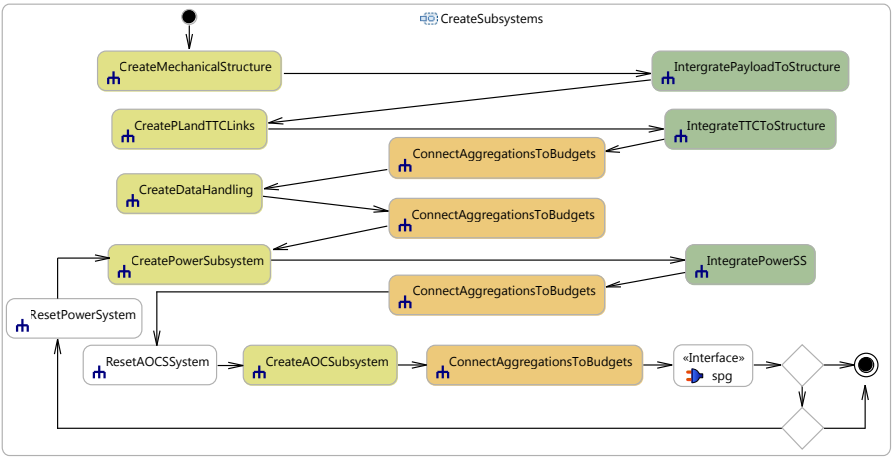


Abbildung 4.43: Einbau der Subsysteme in den FireSat

In den gelben Subaktivitäten in Abb. 4.43 werden jeweils die Subsystem-Entwurfssprachen aufgerufen. Die Subsystem-Entwurfssprachen sind so aufgebaut, dass sie sich selbst in den bestehenden Entwurf einfügen können. Alle missionsspezifischen Anpassungen zur Integration der Subsysteme werden in den grünen Subaktivitäten ausgeführt. Dort werden die Subsysteme zum Beispiel in die Struktur des Satelliten integriert. In den orangen Subaktivitäten werden die eingebauten Subsysteme an die Systembudgets angeschlossen.

Als erstes Subsystem wird die Struktur eingebaut, da alle anderen Subsysteme damit gekoppelt sind. In der zweiten Subaktivität wird dann die schon vorhandene Nutzlastgeometrie in die Struktur integriert. Dann folgt der Einbau des Kommunikationssystems, da dieses hauptsächlich von den Übertragungsanforderungen der Nutzlast abhängig ist. Das System des Bordcomputers wird auf Basis der Nutzlast-Datenmenge ausgelegt. Das Energieversorgungssystem bekommt vom *PowerBudget* den aktuellen Leistungsbedarf übertragen und legt damit wie in Abschnitt 4.2.4 gezeigt, die Solarpanele und den Rest des Systems aus. Das Lageregelungssystem baut abhängig vom Trägheitsmoment des Satelliten unterschiedlich starke

Aktuatoren ein. Daher können unterschiedliche Leistungsanforderungen von diesem System ausgehen.

Da der Einbau des Lageregelungssystems nach dem Einbau des Energieversorgungssystems stattfindet, muss im Anschluss daran überprüft werden, ob das Energieversorgungssystem genügend Leistung liefert. Durch den Iterationszyklus kann die Konsistenz des Entwurfs wieder hergestellt werden. Der erste DecisionNode in Abb. 4.43 unten rechts überprüft, ob das Energieversorgungssystem nach wie vor genügend Leistung liefert. Im zweiten DecisionNode darunter wird ein Abbruchkriterium überprüft, falls das Energieversorgungssystem die Anforderung nicht erfüllen kann.

4.3.2 Analyse Nutzlast

Nach dem die erste Zeile des Aktivitätsdiagramms in Abb. 4.42 ausgeführt ist, sind die Instanzen zur Berechnung der Nutzlast und des Orbit bereits vollständig aufgebaut. Daher kann man nach dieser Zeile eine Analyse der Nutzlast in Bezug auf die Anforderungen durchführen.

In einer Entwurfssprache kann zu jedem Zeitpunkt während des Entwurfs der Lösungspfad für das gesamte Gleichungssystem des Entwurfs generiert werden. Dieser Lösungspfad enthält für jede Variable im Entwurf die Information, durch welche Gleichung in Abhängigkeit welcher weiteren Variablen sie gelöst wird. Wenn man nun in das Gleichungssystem keine Werte einsetzt, sondern das Gleichungssystem analytisch löst, kann man durch Rückeinsetzen für jede Variable eine Funktion nur in Abhängigkeit der Randbedingungen erhalten. Dies soll nun anhand von Abb. 4.44 erläutert werden.

In Abb. 4.44 ist der Lösungspfad der ganz unten stehenden Variable `payload.mass`, der Nutzlastmasse, dargestellt. Die Masse der Nutzlast wird über die in [70, Seite 296] angegebenen

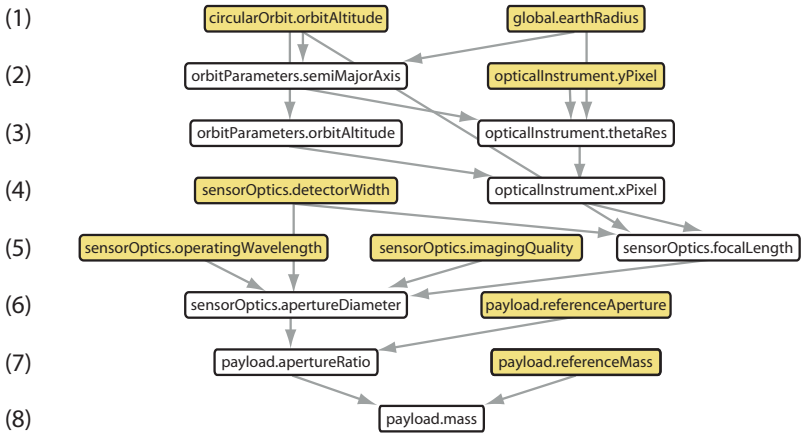


Abbildung 4.44: Lösungspfad der Nutzlastmasse

Näherungen auf der Basis existierender Nutzlasten berechnet. Mittels der Näherungen wird eine Referenzmasse (`referenceMass`) über ein Verhältnis des Referenzaperturdurchmessers (`referenceAperture`) zu dem Aperturdurchmesser der Nutzlast (`apertureDiameter`) auf den für diesen Entwurf gültigen Wert skaliert.

Ausgehend von dem Lösungspfad in Abb. 4.44 kann man durch Rückeinsetzen für jede weiß unterlegte Variable eine Funktion erhalten, die nur von den gelb unterlegten Randbedingungen abhängt. Dies wird am Beispiel der untersten Variable in Abb. 4.44, der Nutzlastmasse gezeigt.

Für das Rückeinsetzen wird im ersten Schritt in die Gleichung zur Berechnung der Nutzlastmasse die Variable der Referenzmasse `payload.referenceMass` eingesetzt. Für den zweiten Pfeil von `payload.apertureRatio` wird die Gleichung zur Berechnung des Apertur-Verhältnisses nach diesem umgestellt und in die Gleichung zur Berechnung der Nutzlastmasse eingesetzt. Man erhält eine Gleichung, die von `sensorOptics.apertureDiameter`, `payload.referenceAperture` und `payload.referenceMass` abhängt. Dann wird für den Wert `sensorOptics.apertureDiameter` wieder die Berechnungsgleichung nach diesem Wert umgestellt und in die vorhandene Gleichung eingesetzt. Diese Schritte werden entlang des gesamten Lösungspfades bis zur obersten Ebene wiederholt. Das Ergebnis des Rückeinsetzens ist Gleichung (4.26).

$$\begin{aligned} \text{payload.mass} = \; & 29.053 \cdot \text{circularOrbit.orbitAltitude}^3 \cdot \text{opticalInstrument.yPixel}^{-3} \\ & \cdot \text{sensorOptics.operatingWavelength}^3 \cdot \text{payload.referenceMass} \\ & \cdot \text{sensorOptics.imagingQuality}^3 \cdot \text{payload.referenceAperture}^{-3} \end{aligned} \tag{4.26}$$

Die Gleichung (4.26) enthält nur sechs der acht in Abb. 4.44 gelb markierten Variablen. Dies liegt daran, dass sich manche der Variablen herauskürzen. Zum einen ist dies der Erdradius, der nur für die Umrechnung der Orbithöhe in die große Halbachse des Orbits benötigt wird. Zum andern ist es die Größe des Detektors, die an zwei Stellen eingeht und in Wirklichkeit keinen Einfluss auf die Masse hat. Durch die analytische Behandlung des Entwurfswissens bleibt die Semantik der Gleichungen erhalten und so können sich die Werte herauskürzen, die für die Berechnung irrelevant sind.

Um den Einfluss der mehrdimensionalen Randbedingungen auf die Zielgröße besser einschätzen zu können, kann man jeweils zwei Randbedingungen über der Zielgröße auftragen. Durch die Form der entstehenden Zustandsfläche kann man den Einfluss der beiden Variablen auf die Zielgröße gut vergleichen. In Abb. 4.45 ist die Masse der Nutzlast über der Größe eines Bodenpixels und der Wellenlänge aufgetragen.

Die Masse der Nutzlast in Abb. 4.45 wächst mit steigender Auflösung sowie mit steigender Wellenlänge. Dies erklärt sich Folgendermaßen: Die Masse der Nutzlast wird über eine Korrelation mit bekannten Instrumenten in Abhängigkeit des Aperturdurchmessers berechnet. Der Aperturdurchmesser steigt mit wachsender Auflösung und mit wachsender Wellenlänge kontinuierlich an. Aus dieser Korrelation ergibt sich die Form der dargestellten Zustandsfläche. Ein weiterer Aspekt der Zustandsfläche in Abb. 4.45 ist, dass die Abhängigkeit der Masse von der Wellenlänge mit steigender Auflösung ansteigt. Die Sensitivität des Entwurfs auf die

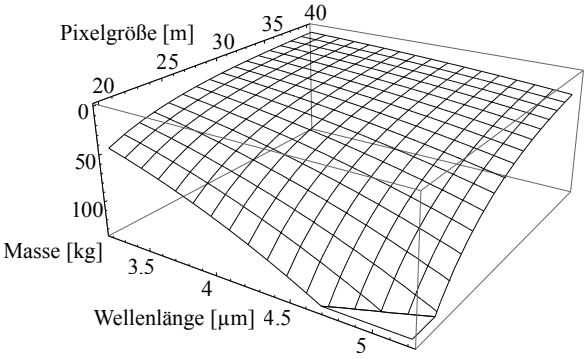
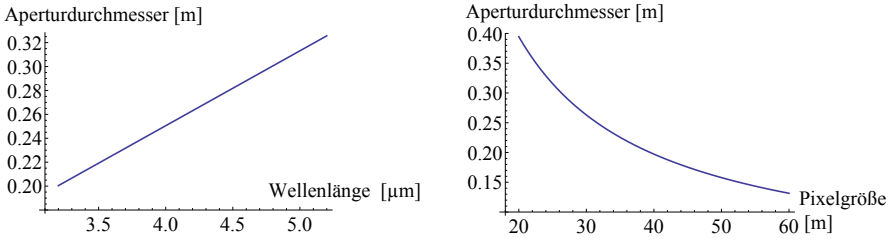


Abbildung 4.45: Masse der Nutzlast

Wellenlänge hängt also von der Auflösung ab. Um die Größe des Teleskops abschätzen zu können, ist der Aperturdurchmesser in Abb. 4.46(a) über der Wellenlänge und in Abb. 4.46(b) über der Größe eines Bodenpixels aufgetragen.



(a) Aperturdurchmesser über Wellenlänge

(b) Aperturdurchmesser über Pixelgröße

Abbildung 4.46: Aperturdurchmesser des Teleskops

Man kann sehen, dass der Aperturdurchmesser linear mit der Wellenlänge ansteigt. Bei der Pixelgröße ergibt sich hingegen ein exponentielles Wachstum hin zu höheren Auflösungen. So hat ein Teleskop mit 30m Bodenauflösung noch etwa 27cm Durchmesser, ein Teleskop mit 20m Bodenauflösung hat hingegen schon über 40cm Durchmesser.

Die gezeigten Kurven und Zustandsfläche gelten unter konstanten Randbedingungen im Auslegungspunkt. Die Entwurfsparameter der Nutzlast haben jedoch einen Einfluss auf den gesamten Systementwurf. Um diesen Einfluss zu untersuchen, wird nun die Analyse auf Systemebene durchgeführt.

4.3.3 Analyse Gesamtsystem

Die Nutzlast ist nur ein Subsystem des Satelliten. Möchte man die Masse des Gesamtsystems untersuchen, so kommen die Beiträge der anderen Subsysteme zu der Gesamtmasse hinzu. In der FireSat Entwurfssprache werden diese Beiträge nach dem Ausführen der Aktivität in Abb. 4.42 berechnet. In Abb. 4.47 ist der Lösungspfad der Gesamtmasse des Satelliten eine Ebene tief dargestellt.

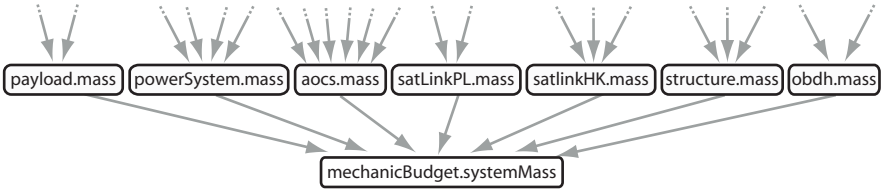
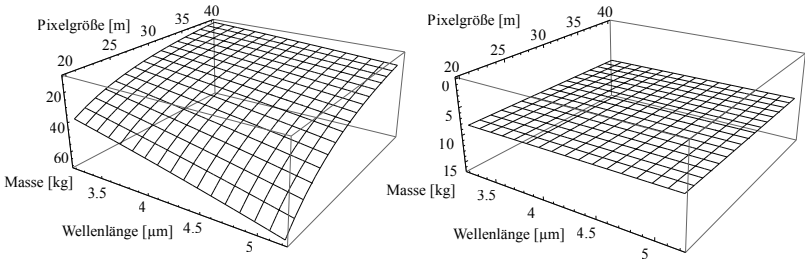


Abbildung 4.47: Lösungspfad der Systemmasse

Man sieht, dass sich die Gesamtmasse des Satelliten aus den Massen der Nutzlast und der einzelnen Subsysteme zusammensetzt. Durch die automatische Analyse des Gleichungssystems kann man die rückeingesetzten Gleichungen für alle Teilbeiträge über die Randbedingungen auftragen. In den Abbildungen 4.48 - 4.50 sind die weiteren Beiträge neben der Nutzlastmasse zur Gesamtmasse des Satelliten über der Größe eines Bodenpixels (`yPixel`) und der Wellenlänge des Instruments (`operatingWavelength`) aufgetragen. In Abb. 4.48 ist die Masse des Energieversorgungssystems und die Masse des Lageregelungssystems aufgetragen.



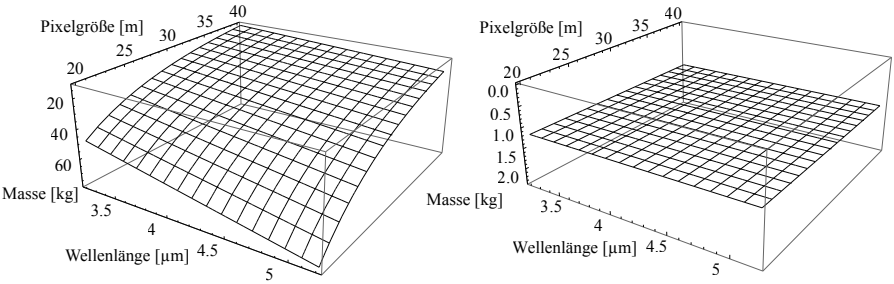
a) Energieversorgungssystem b) Lageregelungssystem

Abbildung 4.48: Zustandsflächen zweier Subsysteme

Der Energieverbrauch der Nutzlast wird über eine Korrelation mit dem Aperturdurchmesser bestimmt. Daher ergibt sich für den Energieverbrauch der Nutzlast eine ähnliche Zunahme über die Auflösung und die Wellenlänge wie bei der Nutzlastmasse. Da die Masse des Energieversorgungssystems von dem Energieverbrauch der Nutzlast abhängt, wächst die Masse des Energieversorgungssystems ebenfalls über steigende Auflösung und steigende Wellenlänge.

Die Masse des Lageregelungssystems in Abb. 4.48 b) ist in dieser Betrachtung konstant über den Entwurfsbereich, da die Komponenten regelbasiert aus einem Katalog an Komponenten ausgesucht werden. Die Masse ist also nur scheinbar unabhängig von den aufgetragenen Auslegungsgrößen, da die Abhängigkeiten in den Regeln der Entwurfssprache formuliert sind. Die aufgezeichneten Flächen gelten eventuell nur für den Auslegungspunkt bei 30m und $4.2\mu m$.

In Abb. 4.49 sind die Massen des Kommunikationssystems für Nutzlastdaten und Telemetriedaten dargestellt. Die Masse des Kommunikationssystem für die Nutzlastdaten in Abb. 4.49



c) Kommunikationsystem Nutzlastdaten d) Kommunikationsystem Telemetrie

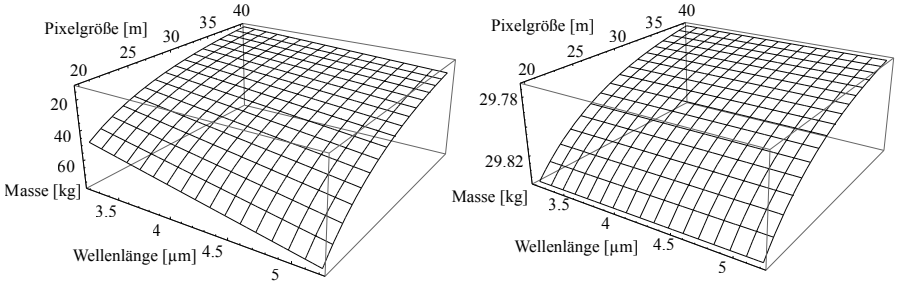
Abbildung 4.49: Zustandsflächen zweier Subsysteme

d) hängt über die Datenrate der Nutzlast von den beiden Randbedingungen ab. Je kleiner die Größe eines Bodenpixels ist, desto mehr Pixel benötigt man um die gleiche Fläche auf dem Boden aufzunehmen. Daher wachsen die Datenrate und damit das zuständige Kommunikationssystem mit sinkender Pixelgröße. Die Datenrate ist jedoch unabhängig von der Wellenlänge des Instruments. Die Abhängigkeit von der Wellenlänge ergibt sich über die Abhängigkeit der Antennengröße von der Struktur. Die Größe der Antenne (in diesem Fall mit Parabolspiegel) wird durch die Breite der Struktur bestimmt. Da die Struktur jedoch wiederum von der Größe des Instruments abhängt und in die Berechnung der Größe des Instruments die Wellenlänge eingeht, hängt die Masse des Kommunikationssystems auch von der Wellenlänge ab.

Das Kommunikationssystem für Telemetriedaten hängt lediglich von der Menge der produzierten Telemetriedaten der Subsystem-Komponenten ab. Daher ändert sich die Masse des Systems in dieser rein parametrischen Betrachtung nicht. Die Masse kann sich nur dann ändern, wenn über Entwurfsregeln mehr oder weniger Komponenten eingebaut werden.

Die Masse der Struktur in 4.50 e) berechnet sich aus den einzelnen Bauteilen der Satellitenstruktur. Die Größe des Satelliten wird durch die Fokallänge des Instruments bestimmt und hängt daher auf ähnliche Art und Weise wie die Nutzlastmasse von den beiden Parametern `yPixel` und `operatingWavelenght` ab.

Die Masse des Datenverarbeitungssystem in Abb. 4.50 hängt über die benötigte Speicherkapazität von der Pixelgröße ab, jedoch nur in einem sehr begrenzten Umfang. Die Massenänderung des Speichers ist im Vergleich zur Masse des Gesamtsystems gering.



e) Struktur f) Datenverarbeitungssystem

Abbildung 4.50: Zustandsflächen zweier Subsysteme

Für die Masse des Lageregelungssystems und die Masse des Kommunikationssystems für Telemetriedaten kann sich jedoch eine Massenänderung ergeben, wenn die Entwurfssprache unter den geänderten Randbedingungen neu ausgeführt wird. Die in Abb. 4.48 - 4.50 gezeigten Zustandsflächen gelten für den Auslegungspunkt bei einer Auflösung von 30m und einer Wellenlänge von 4.2nm. Die Massen außerhalb dieses Punktes sind nur Richtwerte, da durch eine Regel im Entwurfsablauf jederzeit Sprünge in diesen Zustandsflächen entstehen können.

Die Masse des Gesamtsystems ergibt sich aus der Addition der Teilsystemmassen. Die Zustandsfläche in Abb. 4.51 ist eine Überlagerung aller Teilbeiträge aus Abb. 4.48 - 4.50. Diese

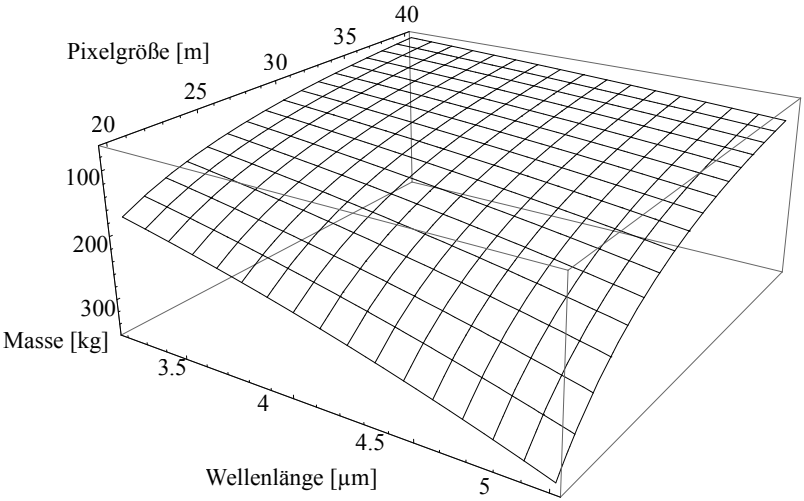


Abbildung 4.51: Zustandsfläche des Gesamtsystems über Wellenlänge

Zustandsfläche stellt die Masse des Gesamtsystems über einen Entwurfsbereich dar. Für die Auslegung eines Satelliten ist es sehr interessant zu sehen, bis zu welchen Randbedingungen man das System zum Beispiel unter 100 kg auslegen kann. Da es jedoch noch weitere Randbedingungen gibt, die sich im Lauf des Entwurfs ändern können, ist es wichtig, die Relationen der Einflüsse zu berücksichtigen.

In Abb. 4.52 ist der Einfluss der Orbithöhe (`orbitAltitude`) und der Größe eines Bodenpixels (`yPixel`) auf die Systemmasse dargestellt. Damit kann der relative Einfluss der Orbithöhe gut mit dem Einfluss der Wellenlänge aus Abb. 4.51 verglichen werden.

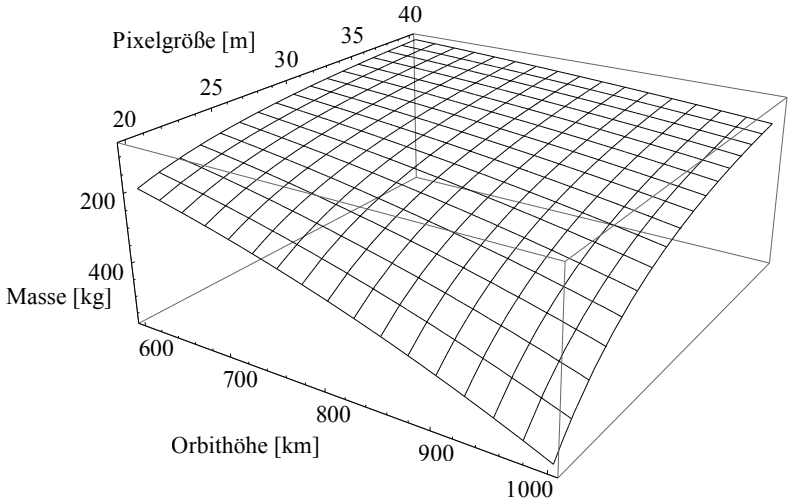


Abbildung 4.52: Zustandsfläche des Gesamtsystems über Orbithöhe

In Abb. 4.52 kann man erkennen, dass die Pixelgröße in dem aufgetragenen Wertebereich einen stärkeren Einfluss auf die Systemmasse hat als die Orbithöhe. Es ist jedoch auch zu erkennen, dass mit geringerer Pixelgröße der Einfluss der Orbithöhe ansteigt, da die Steigung in dieser Richtung größer wird. Dies ist eine interessante Erkenntnis für den Entwurf des Systems, da es einen Hinweis darauf gibt, dass der Entwurf mit steigender Auflösung eine stärkere Kopplung zum Orbit bekommt. Der Einfluss des Orbits ist über diesen Wertebereich auch höher als der Einfluss der Wellenlänge in Abb. 4.51, da die Systemmasse auf über 400 kg ansteigt.

Da das Gleichungssystem des Satelliten jedoch von den Lösungsprinzipien der eingesetzten Komponenten abhängt, sind die Zustandsflächen aus Abb. 4.51 und Abb. 4.52 ebenfalls nur für dieses spezielle Set an Lösunsgprinzipien gültig. Da in einer Entwurfssprache die Lösungsprinzipien austauschbar hinterlegt sind, kann folgende Frage gestellt werden: Wie sieht die Zustandsfläche aus, wenn verschiedene Lösungsprinzipien eingesetzt werden?

4.3.4 Analyse von Systemtopologien

Für die Analyse von verschiedenen Kombinationen von Lösungsprinzipien, sogenannten Systemtopologien, wird die Entwurfssprache über den untersuchten Wertebereich hinweg für jeden Auslegungspunkt neu ausgeführt. Damit werden alle Entscheidungsknoten im Entwurfsablauf berücksichtigt. So ergeben sich durch die eingebauten Entwurfsregeln durch verschiedene Randbedinungen verschiedene Systemtopologien.

Im Energieversorgungssubsystem aus Abschnitt 4.2.4 wird gezeigt, wie auf Entscheidungsknoten basierend die Anzahl der Solarpanele, die an den Satellit angebaut werden, verändert wird. Dies geschieht, wenn die Leistungsanforderung der restlichen Systeme zu groß wird, um sie mit der gegebenen Konfiguration zu erfüllen. Aus einer solchen Entscheidung im Entwurfsablauf ergeben sich Änderungen der Systemtopologie und damit auch des Gleichungssystems.

Zunächst sollen die Änderungen aufgrund solcher im voraus bekannter Entscheidungen untersucht werden. Dafür wird ein Satellit mit einer Hornantenne im Kommunikationssystem ausgelegt. In Abb. 4.53 ist die Masse des Satelliten nach der funktionalen Auslegung über der Wellenlänge (`operatingWavelength`) und der Größe eines Bodenpixels (`y-Pixel`) aufgetragen. Jeder Punkt in diesem Plot entspricht einem vollständig funktional ausgelegten Satelliten. Das Gleichungssystem des Satelliten enthält zu diesem Entwurfszeitpunkt 1241

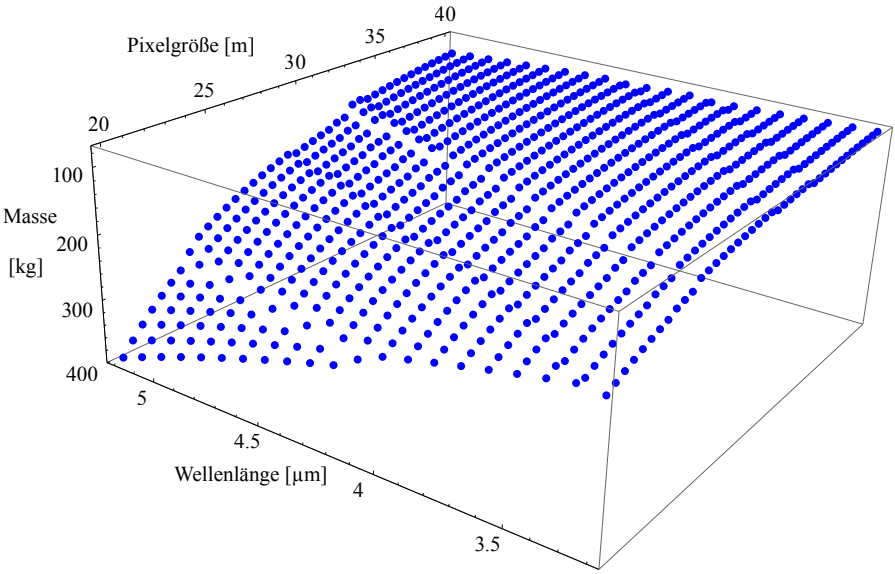


Abbildung 4.53: Masse der Systemtopologie mit Hornantenne

Gleichungen die durch Mathematica in 3,9 Sekunden gelöst werden. Die Ausrichtung der Grafik in Abb. 4.53 ist so gewählt, dass man die Stufen in der Pareto-Fläche des Entwurfs gut erkennen kann. Daher ist die Achse der Wellenlänge im Vergleich zu Abb. 4.51 andersherum aufgetragen. Man kann jedoch die ähnliche Grundform der Zustandsfläche aus Abb. 4.51 in der Pareto-Fläche erkennen.

Die Stufen zwischen den flächigen Bereichen stellen die Begrenzung der Gültigkeitsbereiche einer Systemtopologie und damit eines Gleichungssystems dar. Diese Stufen ergeben sich aus den verschiedenen Entscheidungsknoten im gesamten Entwurfsablauf. Bei einer solchen Stufe hat der Entwurfsablauf einen anderen Verlauf genommen als noch einen Schritt vorher. Es kann zum Beispiel aufgrund einer höheren Leistungsanforderung eine bessere Solarzelle im Energieversorgungssystem verwendet werden, oder das Lageregelungssystem baut aufgrund eines höheren Trägheitsmoments stärkere Aktuatoren ein. Es ist auch möglich, dass sich an einer Stufe mehrere Subsysteme gleichzeitig in ihrer Zusammensetzung ändern, da es zu kaskadierenden Effekten kommt. Dies ist zum Beispiel der Fall wenn die stärkeren Aktuatoren des Lageregelungssystems auch ein größeres Energiesystem notwendig machen.

Sind die in einer bestimmten Situation geltenden Entscheidungsregeln nicht von vorneherein klar, so kann man verschiedene Kombinationen von Lösungsprinzipien miteinander vergleichen. Für die in Abschnitt 4.1 vorgestellten Alternativen des Kommunikationssystems kann man zum Beispiel im Vorhinein keine Aussage über das „beste“ System treffen, da die Masse und der Energieverbrauch gekoppelt sind und sich somit Auswirkungen auf das Gesamtsystem ergeben. Daher wird das Gesamtsystem mit den verschiedenen Varianten des Kommunikationssystems ausgelegt und der Einfluss auf die Gesamtmasse gemessen. In Abb. 4.54 ist die Grafik von Abb. 4.53 um drei weitere Systemtopologien mit jeweils unterschiedlichen Kommunikationssystemen ergänzt.

Die blauen Punkte in Abb. 4.54 stellen wie in Abb. 4.53 einen Satelliten dar, dessen Kommunikationssystem mit einer Hornantenne und einem Transmitter ausgeführt ist. Das durch die grünen Punkte dargestellte System hat zusätzlich zum Transmitter eine Wanderfeldröhre eingebaut. Die gelben Punkte repräsentieren ein System mit Parabolantenne und Transmitter und die roten Punkte stellen die Variante mit einer zusätzlichen Wanderfeldröhre dar.

An jedem Punkt in Abb. 4.54 sind also immer vier Systemtopologien und damit vier farbige Kreise an einer Koordinate überlagert. Das gesamte Diagramm enthält damit 3360 verschiedene, vollständig analytisch ausgelegte Satelliten. Die jeweils leichteste Systemtopologie ist dabei am besten sichtbar, denn der zugehörige Punkt liegt dann etwas weiter oben. Die alternativen Kombinationen sind darunter verdeckt. Man kann erkennen, dass die in Abb. 4.53 vorgestellten Sprünge in den Pareto-Flächen auch hier auftauchen. Interessant ist, dass sich die dominierenden Systemtopologien an den Sprüngen über verschiedene Parameterbereiche hinweg abwechseln. Um die verschiedenen Entwurfsbereiche klarer darzustellen, ist in Abb. 4.55 die Grafik aus Abb. 4.54 noch einmal als Projektion von der massenminimalen Seite (also der negativen Massenachse) dargestellt.

In Abb. 4.55 ist zu erkennen, dass die rote und die gelbe Systemtopologie im gewählten Entwurfsbereich vorherrschen. Das bedeutet, dass ein Satellit mit Parabolantenne und entweder

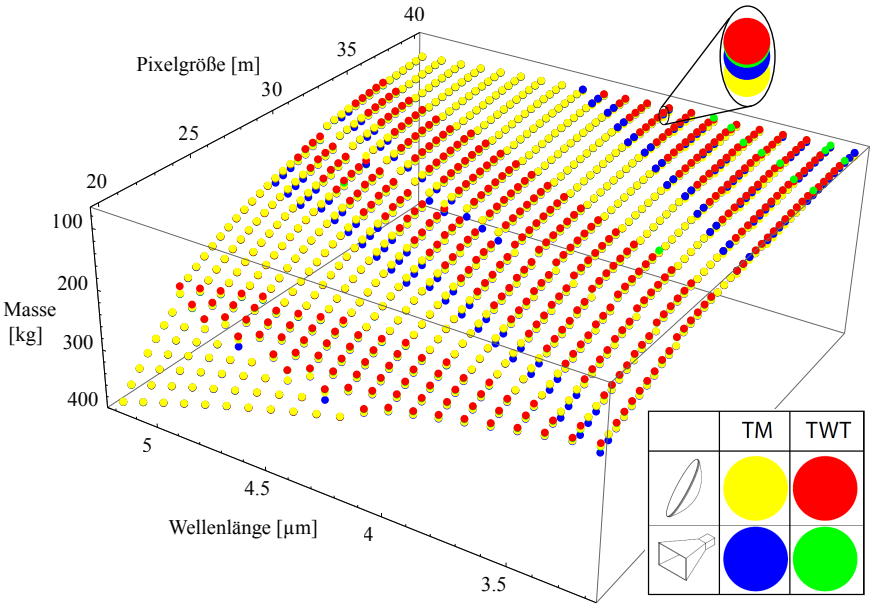


Abbildung 4.54: Masse von Systemtopologien im Vergleich

mit oder ohne Wanderfeldröhre am leichtesten ist. Es gibt jedoch auch Bereiche, in denen die Systemtopologien mit Hornantenne hervortreten.

Die Wechsel in dem rein rot-gelben Bereich des Entwurfsraums finden durch den etwas geringeren Energiebedarf des Systems mit TWT (der *TWTAmplifier* aus Abb. 4.8) statt. Die beiden Systeme haben einen ähnlichen Energieverbrauch. Wenn jedoch der Energiebedarf der Nutzlast steigt (zu kleinen Pixel-Werten hin), dann gibt es Stellen, an welchen die Leistung der Energieversorgung bei einem Satelliten mit TWT gerade noch ausreicht. Bei einem Satelliten ohne Wanderfeldröhre hat aufgrund der leicht höheren Leistungsanforderung jedoch bereits ein Wechsel im Energieversorgungssystem stattgefunden. Dieser Wechsel führt zu einem schwereren Energieversorgungssystem (Batterie, PowerConverter etc.) und damit auch zu einem höheren Gesamtgewicht. In manchen Fällen werden durch das höhere Gewicht im Energieversorgungssystem auch größere Drallräder oder Magnettorquer im Lageregelungssystem notwendig, wodurch sich der Effekt noch verstärkt. Erst wenn auch bei einem Satelliten mit TWT ein größeres Energieversorgungssystem notwendig wird, ist der Satellit ohne TWT wieder leichter.

Um den Entwurfsraum im oberen rechten Eck zu erklären, werden nun die einzelnen Systemtopologien in diesem Bereich dargestellt. In Abb. 4.56 ist die Masse des Gesamtsystems mit einer Parabolantenne und einem Transmitter im Kommunikationssystem aufgetragen.

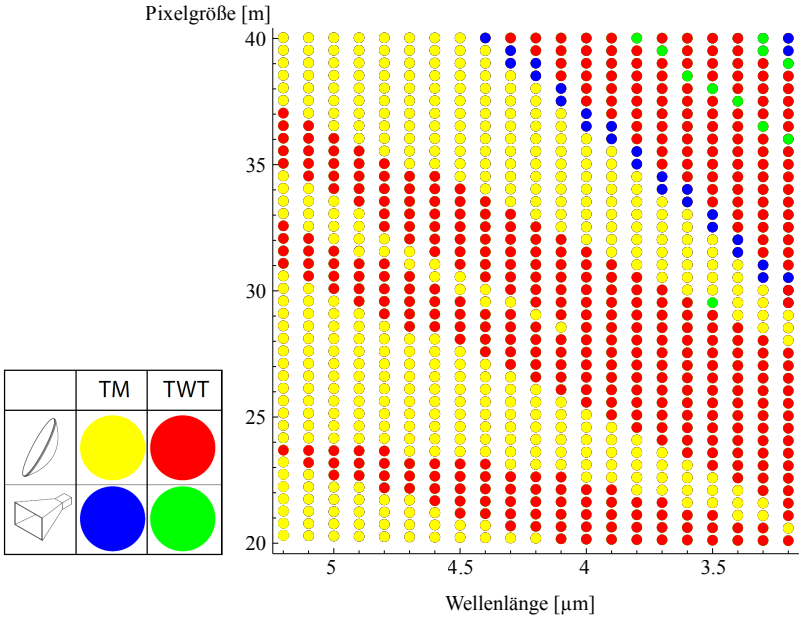


Abbildung 4.55: Entwurfsbereiche der Systemtopologien

Die Masse des Gesamtsystems steigt mit sinkender Pixelgröße und steigt mit steigender Wellenlänge. Es gibt jedoch Stellen, an welchen die Systemmasse sprunghaft ansteigt. In Abb. 4.56 ist der Wechsel von einem Satellit mit zwei klappbaren Panelen auf einen Satellit mit vier klappbaren Panelen farblich dargestellt. Im unteren hellgelben Bereich hat der Satellit zwei, im oberen dunkelgelben Bereich vier klappbare Panele.

Der zweite Sprung in der dunkelgelben Pareto-Fläche ist dann dem Wechsel zu Solarzellen mit einer höheren Effizienz zuzuschreiben. Diese Wechsel sorgen für einen Massenanstieg im Energieversorgungssystem und damit für eine steigende Masse des Gesamtsystems. In Abb. 4.57 ist derselbe Ausschnitt aus dem Entwurfsraum auch für die drei anderen Systemtopologien des Kommunikationssystems dargestellt.

Der Wechsel auf vier Panele verschiebt sich von blau über rot nach grün ins obere rechte Eck. Dies bedeutet, dass die Systemvarianten mit Hornantenne (blau), mit Parabolantenne und Wanderfeldröhre (rot) und mit Hornantenne und Wanderfeldröhre (grün), jeweils effizienter arbeiten und daher erst später zu vier Panelen wechseln. Durch die Überlagerung der vier Grafiken aus Abb. 4.56 und 4.57 ergibt sich das obere rechte Eck in der Abb. 4.55.

Die in diesem Abschnitt vorgestellten Mittel zur Synthese und Analyse komplexer Systementwürfe bieten eine breite Basis für die Untersuchung verschiedener Systemtopologien.

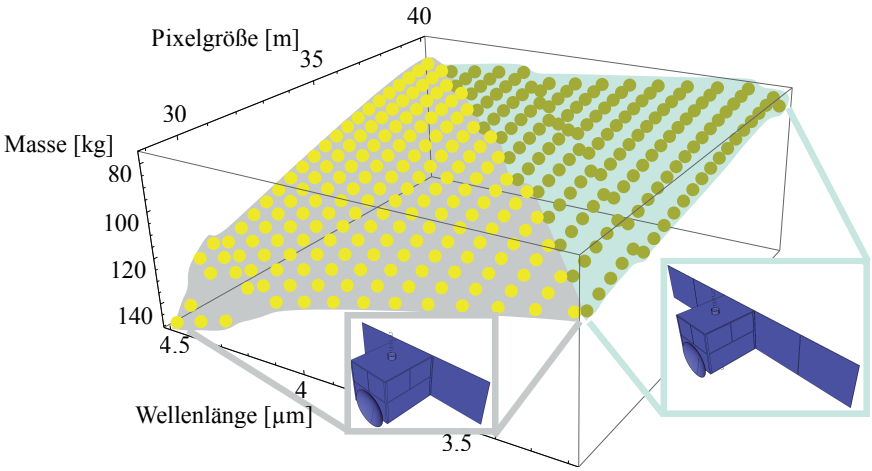


Abbildung 4.56: Wechsel in einer Systemtopologie

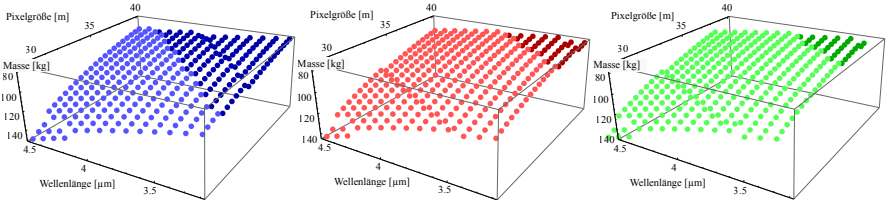


Abbildung 4.57: Masse dreier Systemtopologien

Durch die automatische Synthese können große Gleichungssysteme untersucht werden, die man wegen des Aufwandes von Hand nicht aufstellen würde. Durch die Mittel einer Entwurfssprache kann der Entwurfsablauf programmiert werden und durch diese Programmierung können Entscheidungsmuster untersucht und hinterfragt werden. Damit kann auf analytischer Basis ein Optimum unter den gegebenen Anforderungen gefunden werden.

Die gezeigten Analysen führen zu einem besseren Verständnis des Entwurfsraumes über die dargestellten Variablen. Da der Entwurfsraum in der vorgestellten Entwurfssprache jedoch sehr viele Dimensionen (Variablen) enthält, ist es ebenfalls aufschlussreich, in einem Arbeitspunkt alle Variablen zu betrachten. Dies wird mit einer Sensitivitätsanalyse erreicht.

4.3.5 Sensitivitätsanalyse

Durch die automatische Erstellung des Gleichungssystems für jede Variable im Entwurf ist es mit dem von Michael Bölling [9] entwickelten Code ebenso auch möglich, alle Variablen nach den Randbedingungen abzuleiten. Daraus resultiert eine sogenannte Heatmap für das Gesamtsystem Satellit. In Abb. 4.58 ist diese Heatmap dargestellt. In dieser Matrix sind nur die für eine funktionale Systemauslegung interessanten Werte enthalten. Alle Zwischenergebnisse und Hilfsvariablen sind aus Platzgründen nicht abgebildet. Für eine genauere Erklärung wie die Werte berechnet werden sei auf [9] verwiesen. Dort finden sich auch weitere Darstellungen und Analysen, in denen alle Variablennamen enthalten sind.

Die Variablen in Abb. 4.58 sind von Hand angeordnet. Die berechneten Variablen des Entwurfs sind zeilenweise aufgetragen, die Randbedingungen spaltenweise. Alle Variablen sind nach Subsystemen gruppiert. Die Anordnung ist dabei so gewählt, dass eine obere Dreiecksmatrix entsteht. Die Farbskala in diesem Beispiel erstreckt sich von weiß (0) über gelb (0.00001) und orange (1) nach rot (10). Im negativen Zahlenraum von weiß (0) über türkis (-0.00001) und blau (-1) nach schwarz(-10). Im ersten Feld links oben sind die Einflüsse der Randbedingungen für das Kommunikationssystem (TTC) (die Spalten) auf die berechneten Variablen des Kommunikationssystem (die Zeilen) in den Farben der Heatmap dargestellt. In den Feldern rechts davon sind die Einflüsse anderer Randbedingungen auf das Kommunikationssystem dargestellt. Das Kommunikationssystem wird von der Nutzlast, dem Orbit und der Mission beeinflusst.

In dem nächsten Zeilenblock sind die Einflüsse auf die Struktur dargestellt. Die Struktur wird nur von den eigenen Randbedingungen, von der Nutzlast und vom Orbit beeinflusst. Auch das On-Board Data Handling (OBDH) wird nur von den eigenen Randbedingungen, der Nutzlast, dem Orbit und der Mission beeinflusst. Das Lageregelungssystem (AOCS) nimmt scheinbar eine Sonderstellung ein, da es keine Abhängigkeit von der Nutzlast und dem Orbit aufweist. Dies ist jedoch nur für das Gleichungssystem richtig. Durch die Entscheidungsknoten im Aktivitätsdiagramm findet auch hier eine Kopplung zur Systemmasse und damit zu nahezu allen Randbedingungen statt.

Das Energieversorgungssystem hängt direkt vom Energiebudget und damit von allen Subsystemen außer der Struktur ab. Die Nutzlast dient in der Auslegung als Vorgabe für alle Subsysteme und hängt nur von sich selbst und dem Orbit ab. Die Mission hat keine Auswirkungen auf die Nutzlast, da das Teleskop nur nach dem Orbit und den Vorgaben für die Nutzlast ausgelegt wird. Der Orbit wiederum hängt nur von sich selbst ab. Die Missionsparameter hängen vom Orbit und den eigenen Randbedingungen ab. In dem untersten Block sind die Systembudgets aufgetragen. Die unteren Zeilen stellen das Massenbudget dar. Hier sieht man eine Abhängigkeit von nahezu allen Randbedingungen. Die Farbwerte sind jedoch sehr hell in dieser Zeile, was auf eine schwache Sensitivität schließen lässt.

Der Entwurf komplexer Systeme beinhaltet nicht nur auf Basis der analytischen Vorauslegungsgleichungen vielfältige Kopplungen. Durch die geometrische Integration aller Komponenten in ein Produkt entstehen weitere Kopplungen, die untersucht und aufgelöst werden müssen. Dafür muss zunächst die Konfiguration des Entwurfs auf geeignete Weise beschrieben werden. Wie dies in einer Entwurfssprache vorgenommen wird, zeigt der folgende Abschnitt.

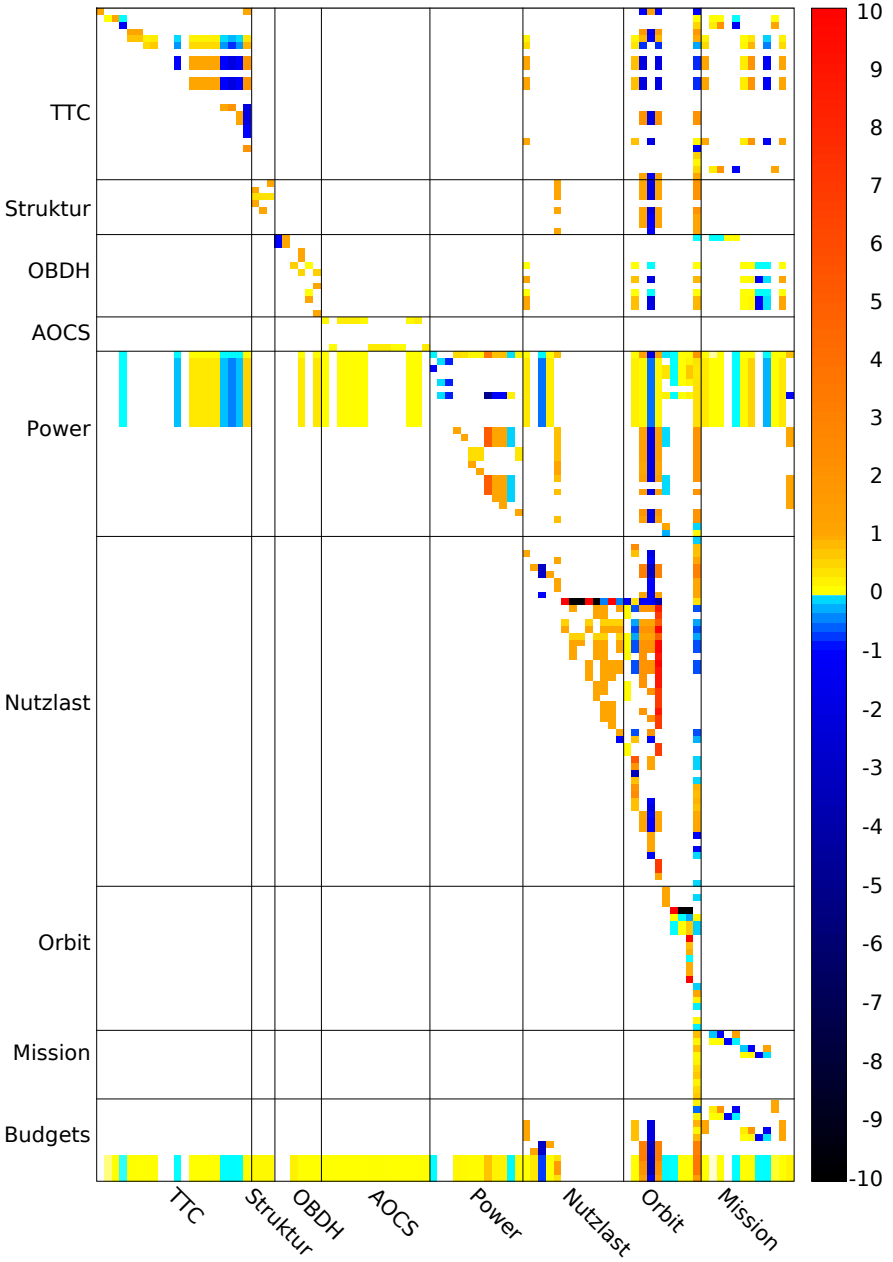


Abbildung 4.58: Heatmap-Darstellung des FireSat nach [9]

4.4 Konfiguration

In der Konfiguration wird die topologische und parametrische Anordnung der Komponenten des Satelliten zueinander definiert. In dieser Arbeit wird mit der Einbindung einer abstrakten Geometriedarstellung die Basis dafür geschaffen. Über die folgenden Abschnitte wird erklärt, wie mit dieser abstrakten Geometrierepräsentation regelbasiert ein Geometriemodell aufgebaut werden kann. Im letzten Abschnitt folgt eine Machbarkeitsdemonstration für einen Algorithmus, der die Anordnung der Komponenten durchführt.

Die Geometriedefinition in dieser Arbeit beruht auf der in [65] entwickelten Entwurfssprache für Geometrie. Die Definition und Bearbeitung der Geometrie soll in diesem Abschnitt anhand der Beispielgeometrie in Abb. 4.59 erläutert werden.

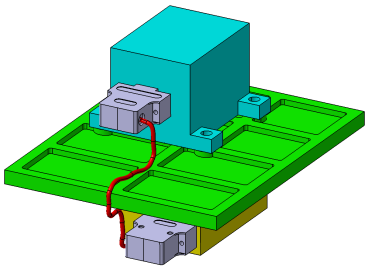


Abbildung 4.59: Beispielgeometrie mit zwei Boxen

Bei dem Beispiel in Abb. 4.59 handelt es sich um einen Ausschnitt aus einem Satellit. Es gibt zwei Elektronikboxen und eine Einbauplatte, auf der diese befestigt werden. Die Boxen sind mit Halterungslaschen versehen und in die Platte sind Versteifungstaschen eingefräst. An den Boxen sind Steckerhüllen angebracht zwischen welchen ein Kabel verläuft.

4.4.1 Geometrie

Grundlage für die Beschreibung der Geometrie eines Gesamtsystems in einer Entwurfssprache ist die räumliche Anordnung der geometrischen Elemente und ihre Beziehung dabei zueinander. Für diesen Zweck gibt es in der Entwurfssprache für Geometrie zwei Klassen, die Komponente (*Component*) und die Position (*Position*). In Abb. 4.60 sind diese Klassen abgebildet.

Die *Component* und die *Position* haben als gemeinsame Oberklasse das *TopologyElement*. Dieses definiert über eine Selbstreferenz `sub` die Verbindungen zwischen *Component*s und *Position*s gegenseitig und untereinander. Eine *Component* kann entweder direkt Geometrieelemente, also *Shape*s enthalten, oder weitere *TopologyElement*s. *Component*s können über *Position*s zueinander positioniert werden.

Eine *Component* kann also eine Baugruppe oder ein Bauteil sein. Hat eine *Component* eine *Shape* als Kind, so ist es ein Bauteil, hat sie weitere *TopologyElement*s als Kinder, so ist es eine

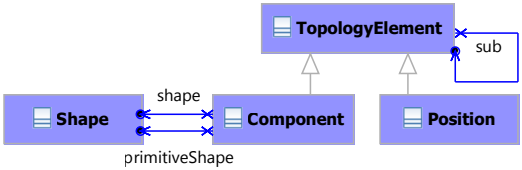


Abbildung 4.60: Topologie und Form in der Entwurfssprache nach [65]

Baugruppe. Diese Modellierung hat den Vorteil, dass ein Bauteil mit einer *Shape* im Verlauf des Entwurfs zu einer Baugruppe umgewandelt werden kann. In diesem Fall wird die ursprüngliche *Shape* eine Ebene tiefer gehängt und durch eine oder mehrere *Component*s ersetzt.

Durch *Component*s, die über *Position*s zueinander positioniert werden, kann ein Strukturbaum in einem CAD-Programm abgebildet werden. Für die Modellierung von Formen werden über einen `shape`-Link *Shape*s an *Component*s angehängt. Der zweite Link von *Component* zu *Shape*, `primitiveShape` dient zur Definition eines (parallelen) vereinfachten Geometriemodells (z.B. für die Thermalsimulation). Für die Modellierung von *Shape*s werden in der in [65] entwickelten Entwurfssprache zwei Grundprinzipien, Constructive Solid Geometry (CSG) und Boundary Representation (B-Rep) verwendet.

In der Constructive Solid Geometry baut man aus Volumenkörpern über boolesche Operationen komplexere Geometrieelemente auf. Die verwendeten Volumenkörper können durch vordefinierte Primitive gegeben sein. In Abb. 4.61 ist ein Ausschnitt des Klassendiagramms mit den derzeit verwendbaren Primitiven abgebildet.

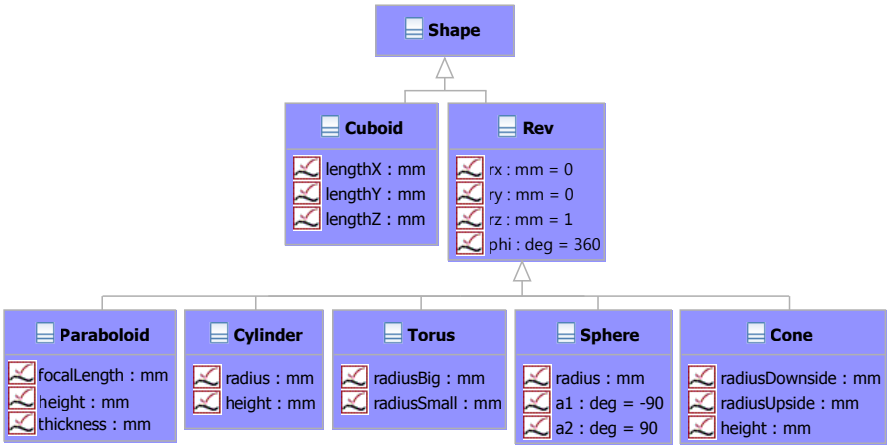


Abbildung 4.61: CSG-Klassen der Entwurfssprache für Geometrie

Alle *Primitive*s können über die Attribute `lx`, `ly` und `lz` im lokalen Koordinatensystem verschoben werden. Ein Quader, genannt *Cuboid*, ist das einzige Primitiv, das keinen Rotationskörper darstellt. Alle anderen Primitive erben von *Rev* die Eigenschaften `rx`, `ry` und `rz` zur Definition einer Rotationsachse sowie `phi` für die Angabe des Rotationswinkels. Unter den Primitiven finden sich Standardköper wie ein Zylinder (*Cylinder*) oder ein *Torus*, aber es können auch speziellere Formen, wie zum Beispiel ein *Paraboloid*, implementiert werden.

Für die weitere Bearbeitung von Volumenkörpern in der CSG-Methode sind in der Entwurfssprache die booleschen Operatoren *Add*, *Cut* und *Common* implementiert. In Abb. 4.62 sind diese Klassen im Diagramm dargestellt.

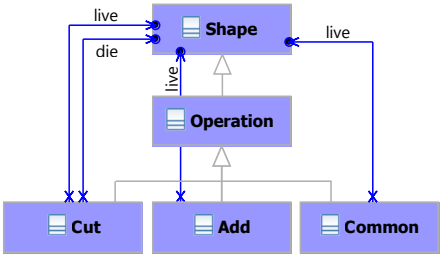


Abbildung 4.62: Operationen in der Entwurfssprache für Geometrie

Die *Operation*s haben jeweils wieder Links zu *Shape*. Ein *Cut* hat eine `live`-*Shape* aus der eine `die`-Shape bei der Operation herausgeschnitten wird. Ein *Add* hat zwei `live`-*Shape*s als Kinder, die addiert werden und bei einem *Common* wird die Schnittmenge der beiden `live`-Kinder bestimmt. Eine vollständige Übersicht der definierten Klassen findet sich in [65].

Die Klassen für das oben gezeigte Beispiel von zwei Komponenten, die auf einer Einbauplatte montiert und verkabelt werden, sind in Abb. 4.63 dargestellt.

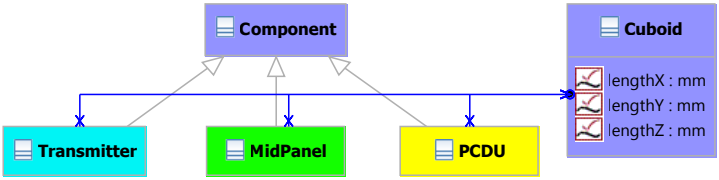


Abbildung 4.63: Klassen des Geometriebeispiels

Die drei Komponenten des Beispiels erben von *Component* und besitzen einen Link zur Klasse *Cuboid*. Damit können sie als Quader repräsentiert werden und diesem Quader Werte für Länge, Breite und Höhe zuweisen. Um diese Klassen zu instanziieren, werden drei Regeln ausgeführt. Abbildung 4.64 zeigt die Abfolge der drei Regeln.

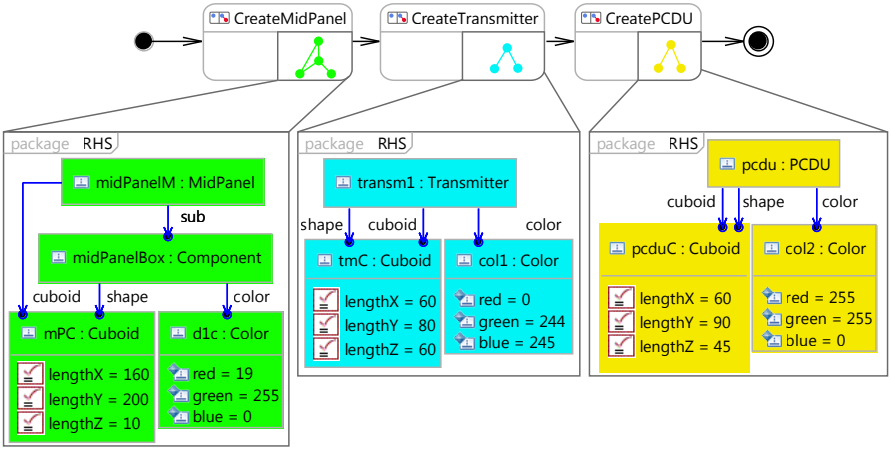


Abbildung 4.64: Aktivität zum Aufbau der Boxen

Im oberen Teil von Abb. 4.64 ist die Aktivität für den Aufbau der Boxen dargestellt. In den Regelknoten sind kleine Darstellungen der Topologie innerhalb der Regel als Graph enthalten. Darunter sind jeweils die rechten Seiten der Regeln in der klassischen UML-Ansicht abgebildet. Die erzeugten Instanzen sind mit Links untereinander verbunden. In der ersten Regel kann man erkennen, dass das *MidPanel* als Baugruppe modelliert wird, in die ein Bauteil mit der *Shape* darunter gehängt wird. Der direkte Link zwischen dem *MidPanel* und dem *Cuboid* dient der Zuordnung der Geometrie zu der funktionalen Panel-Klasse. Die beiden anderen Boxen werden direkt als Bauteil modelliert. Die Werte für Länge (`lenghtX`), Breite (`lenghtY`) und Höhe (`lenghtZ`) des *Cuboid*s werden in der Regel vorgegeben, ebenso wie Werte für die Farben der Boxen. In Abb. 4.65 ist die Geometrie nach der Aktivität in 4.64 dargestellt.

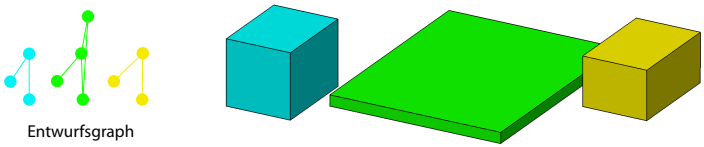


Abbildung 4.65: Entwurfsgraph und Geometrie der Boxen

Die Boxen sind noch nebeneinander dargestellt, da noch keine Zuordnung auf die Einbauplatte stattgefunden hat. Dies wird nun in einem weiteren Schritt durchgeführt. Um die Elektronikboxen auf der Platte zu installieren, wird die Platte mit zwei Einbauebenen versehen. In Abb. 4.66 ist die Regelabfolge zu diesem Vorgang abgebildet.

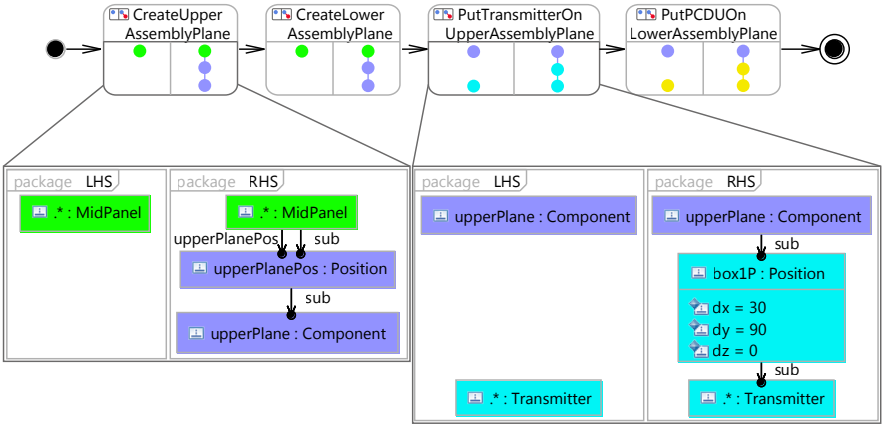


Abbildung 4.66: Regeln für Anordnung der Boxen

Die erste Regel in Abb. 4.66 erzeugt eine Einbauebene auf der Oberseite der Platte, mit positiver z-Richtung nach oben. Auf dieser Einbauebene wird in der zweiten Regel der *Transmitter* topologisch zugeordnet. Die dritte Regel erzeugt eine Einbauebene auf der Unterseite der Platte, mit positiver z-Richtung nach unten. Auf dieser Einbauebene wird die *PCDU* topologisch zugeordnet. Das Ergebnis der Regelabfolge ist in Abb. 4.67 zu sehen. Auf der linken Seite ist an

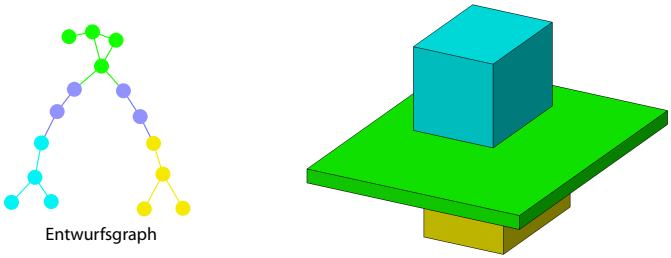


Abbildung 4.67: Angeordnete Boxen

der abstrakten Sicht auf den Entwurfsgraph zu erkennen, dass die Knoten der Elektronikboxen nun unter den Knoten der Einbauebenen hängen. Beide Elektronikboxen sind in z-Richtung auf der Platte angebracht. Die Drehung der unteren Einbauebene um 180° um die x-Achse wirkt sich durch die topologische Zuweisung auch auf die *PCDU* aus.

4.4.2 Halterungen

Um die Bauteile auf der Einbauplatte zu befestigen ist es notwendig, Halterungslaschen an den Bauteilen anzubringen. Für diesen Zweck wurde eine kleine Entwurfssprache erstellt. In Abb. 4.68 ist das Klassendiagramm der Entwurfssprache für Halterungen dargestellt. Die Hal-

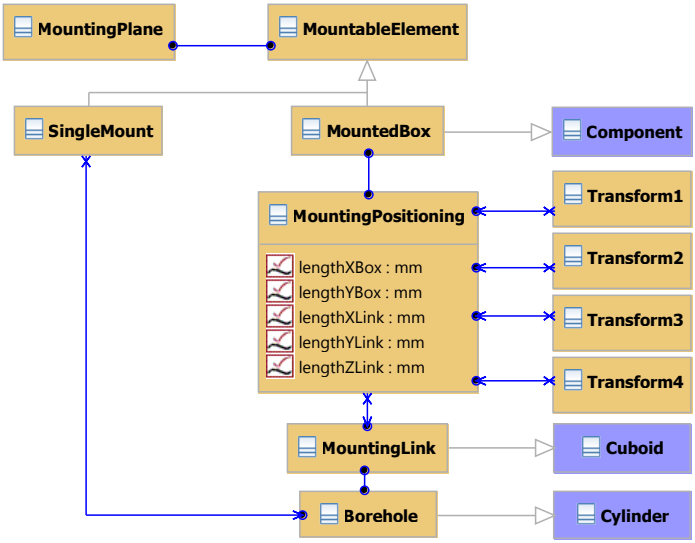


Abbildung 4.68: Klassen der Entwurfssprache für Halterungen

terungslaschen (*MountingLink*) werden über ein *MountingPositioning* an den vier Ecken eines Quaders platziert. Die Laschen (*MountingLink*) selbst sind mit einem Bohrloch (*Borehole*) für eine Schraube versehen. Um eine der Elektronikboxen des Beispiels mit Befestigungslaschen zu versehen, muss die Klasse der Elektronikbox von *MountedBox* erben. In Abb. 4.69 ist die Vererbung an die beiden Boxen dargestellt. Durch diese Vererbung kennzeichnet der Ingenieur, dass die beiden Elektronikboxen mit Halterungslaschen versehen werden sollen. Dies geschieht im Klassendiagramm der jeweiligen Entwurfssprache. Das Anbringen der Halterungslaschen

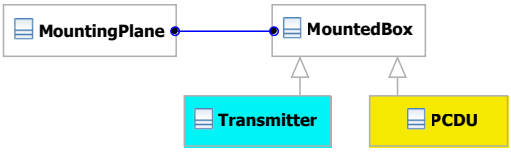


Abbildung 4.69: Einbinden der Halterungen

geschieht hingegen erst, wenn die Aktivtät der Halterungssprache aufgerufen wird. Dadurch ist das Wissen über den Prozess lokal in der Entwurfssprache für Halterungen gespeichert und wird von der Entwurfssprache für dieses Beispiel nur aufgerufen. Ändert sich etwas an den Halterungen und dem Aufbauprozess, so ist dies innerhalb dieser Entwurfssprache gekapselt.

Das Anbringen selbst wird in nur einer Regel modelliert. In Abb. 4.70 ist die Aktivität zum Anbringen der Laschen abgebildet. In der Regel „CreateMounting“ wird eine *MountedBox*

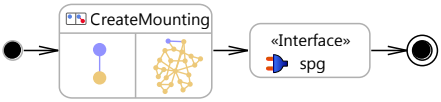


Abbildung 4.70: Aufbau der Halterungen

gesucht, die über einen `shape`-Link mit einem *Cuboid* verbunden ist. Dann wird zwischen die *MountedBox* und den *Cuboid* der Graph für die vier Halterungen eingefügt. Die Halterungen werden also unter zwei Vorbedingungen an einer Elektronikbox angebracht: Zum einen muss der Ingenieur im Klassendiagramm die Vererbungsbeziehung des Bauteils zur *MountedBox* angeben und zum anderen muss die Elektronikbox als einfacher *Cuboid* modelliert sein. In Abb. 4.71 ist auf der linken Seite das Ergebnis der Aktivität in der abstrakten Sicht auf den Entwurfsgraph zu sehen. In den durch die Regel erstellten Instanzen in diesem Entwurfsgraph

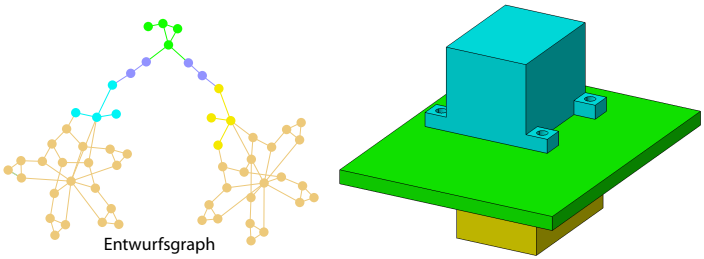


Abbildung 4.71: Beispiel mit Halterungen

sind nun noch keine Werte für die geometrischen Elemente enthalten. Die Längen der *MountingLink*s und der Durchmesser und die Position der *Borehole*s werden erst durch Gleichungen aus der Größe der zu befestigenden *MountedBox* abgeleitet. Aus diesem Grund wird in der Aktivität der Lösungspfadgenerator nach der Regel „CreateMounting“ aufgerufen. Auf der rechten Seite in Abb. 4.71 sind die so entstandenen Halterungslaschen im Geometriemodell erkenntlich. Der blaue *Transmitter* auf der Oberseite der Platte besitzt nun vier Halterungslaschen mit Bohrlöchern. Auch die gelbe *PCDU* auf der Unterseite der Platte besitzt diese Laschen. Damit können beide Boxen auf der Platte befestigt werden.

4.4.3 Packaging

Für die Anordnung (Packaging) wird aufbauend auf Vorarbeiten durch Marc Eheim [17] eine Entwurfssprache auf Basis der Geometriesprache entwickelt. Die Anordnungssprache stellt in der derzeitigen Umsetzung eher einen Nachweis der Machbarkeit für die automatische Anordnung dar. In Abb. 4.72 ist das Klassendiagramm dieser Sprache abgebildet. Das Klas-

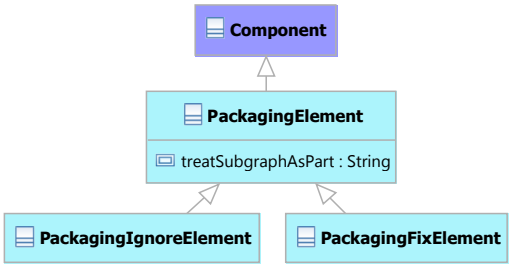


Abbildung 4.72: Klassen der Entwurfssprache für Packaging

sendiagramm ist dementsprechend wenig differenziert. Es gibt *PackagingElements*, welche *Components* der Geometriesprache sind. Alle *PackagingElements* werden für die Erstellung des Geometriemodells für das Packaging verwendet. Soll ein Element für die Überschneidungsanalyse nicht berücksichtigt werden, so ist es ein *PackagingIgnoreElement*. Soll ein Element berücksichtigt werden, sich aber nicht bewegen, wird es als *PackagingFixElement* modelliert. Um das Packaging zu demonstrieren, wird das Beispiel um einige Boxen erweitert und die Topologie des Problems etwas angepasst. In Abb. 4.73 ist die Anfangskonfiguration für dieses Beispiel abgebildet.

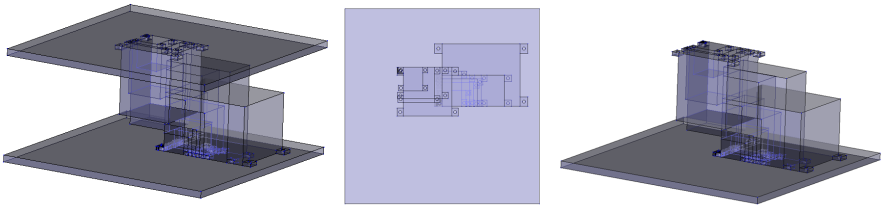


Abbildung 4.73: Anfangskonfiguration des Packaging-Beispiels

Die Konfiguration besteht aus zwei Einbauplatten, auf denen Boxen montiert werden sollen. Die Einbauebenen sind so positioniert, dass die Boxen der beiden Ebenen sich teilweise durchdringen. Dadurch entsteht ein dreidimensionales Problem. Zu Beginn sind die Boxen im Ursprung der jeweiligen Einbauebene platziert und durchdringen sich daher gegenseitig.

Der Algorithmus zum Packaging beginnt mit einer Überschneidungsanalyse der Bauteile. Aus der Überschneidungsanalyse werden für jedes Bauteil, das Überschneidungen mit anderen Bauteilen aufweist, Richtungsvektoren bestimmt. In Abb. 4.74 sind einige Schritte des Packaging mit diesen Richtungsvektoren dargestellt. Das Bauteil wird ein Stück in diese Richtung

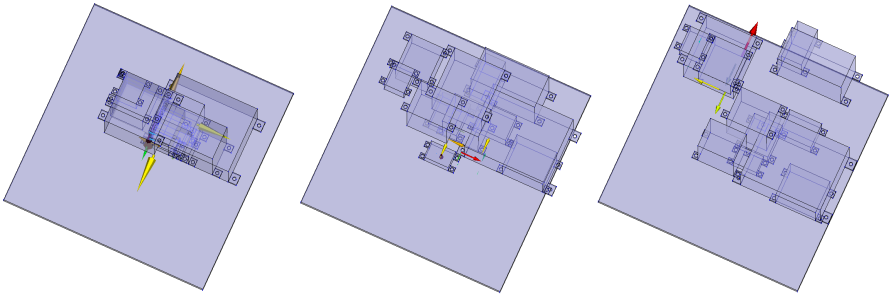


Abbildung 4.74: Sequenz an Anordnungen während des Packagings

bewegt und eine neue Überschneidungsanalyse wird erstellt. Dadurch können die Bauteile in eine überschneidungsfreie Position gebracht werden. In der bisherigen Implementierung des Packaging ist dies die einzige Randbedingung. Es wäre jedoch möglich, auf Basis der vorhandenen Implementierung weitere Randbedingungen, wie zum Beispiel eine Minimierung des Trägheitsmoments oder das Erreichen eines minimalen Volumens, umzusetzen.

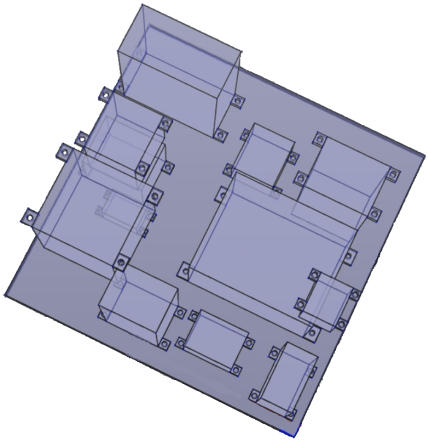


Abbildung 4.75: Fertig angeordnete Boxen

In Abb. 4.75 ist die Einbauebene nach Ausführen der Anordnungssprache abgebildet. Die obere Platte ist ausgeblendet. Man kann erkennen, dass die Bauteile durchdringungsfrei auf den beiden Platten angeordnet sind. Damit hat der Packaging-Algorithmus die erforderten Randbedingungen erfüllt.

Nach der Anordnung der Bauteile im Satelliten ist es notwendig, die aus Aluminium gefertigten Platten durch Ausfräsungen leichter zu machen. Die Ausfräsungen sind sogenannte Taschen. Auch für den Einbau der Taschen wird aufbauend auf durch den Autor betreute Vorarbeiten [44] eine Entwurfssprache erstellt.

4.4.4 Taschen

Die Entwurfssprache für Taschen ermöglicht es, eine *Platte* auf der *Oberseite* und *Unterseite* mit *Tasche*n zu versehen. Das Klassendiagramm der Sprache ist in Abb. 4.76 dargestellt. Die

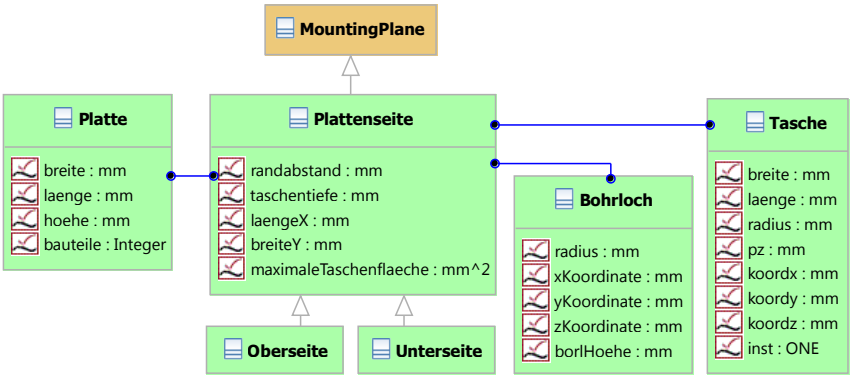


Abbildung 4.76: Klassen für Taschen

Entwurfssprache wurde auf Deutsch entwickelt und erst später in die (in Englisch gehaltene) FireSat-Entwurfssprache eingebunden. Ganz links in Abb. 4.76 ist die *Platte* definiert, welche mit *Tasche*n versehen werden soll. Eine *Platte* hat mehrere *Plattenseite*n. Dies kann zum Beispiel die *Oberseite* und die *Unterseite* sein. Eine *Plattenseite* hat des weiteren *Bohrlöcher* und *Tasche*n. Ein *Bohrloch* gibt an, an welcher Stelle der *Platte* eine Befestigungsbohrung für ein Bauteil sein wird. An dieser Stelle erstellt die Entwurfssprache einen Steg. Die Dicke der Stege und die Größe der Taschen werden nach frei wählbaren Parametern definiert. Da in dieser Entwurfssprache keine Festigkeitsanalyse enthalten ist, findet keine Bewertung der Struktur statt.

Die Verbindung zwischen der Entwurfssprache für Halterungen und der für Taschen wird durch die *MountingPlane* aus der Entwurfssprache für Halterungen hergestellt. Eine *Plattenseite* erbt von *MountingPlane* und kann dadurch mit *MoutingElements* verbunden werden. In Abb. 4.77 ist dargestellt, wie die Entwurfssprache für Taschen in dem Beispiel mit den

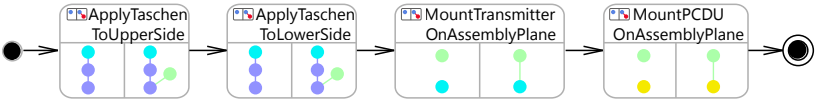


Abbildung 4.77: Integration der Taschen in Beispiel

beiden Elektronikboxen integriert wird. In der ersten Regel wird der *Cuboid* der Einbauplatte gesucht und mit einer neu erstellten Instanz von *Oberseite* verbunden. In der zweiten Regel wird er mit einer Instanz von *Unterseite* verbunden. In der dritten Regel wird der *Transmitter* mit der *Oberseite* über den Link `mountedElement` aus der Entwurfssprache für Halterungen verbunden. In der vierten Regel wird die *PCDU* mit der *Unterseite* ebenfalls über den Link `mountedElement` verbunden. Dann wird die Aktivität „CreateTaschen“ aus der Entwurfssprache für Taschen aufgerufen.

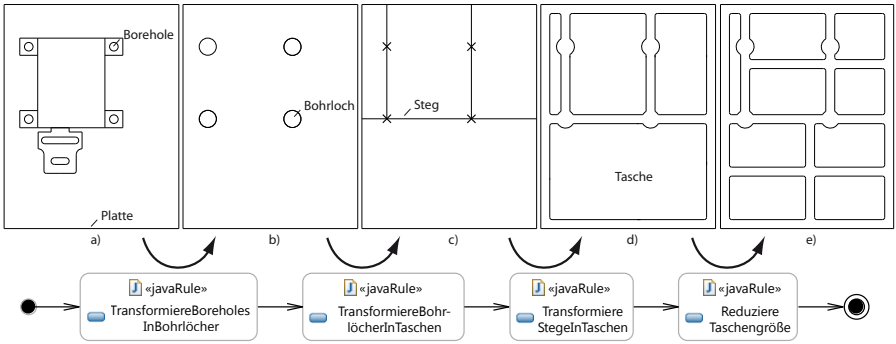


Abbildung 4.78: Aufbau der Taschen

In Abb. 4.78 ist der Vorgang für das Erstellen der Taschen dargestellt. In der Entwurfssprache für Taschen werden zunächst alle *MountedBox*es (d.h. die Elektronikboxen) einer *MountingPlane* (der Ober- bzw. Unterseite) gesucht. Dies entspricht der Darstellung a) in Bild 4.78. Dann werden für alle *MountedBox*es die einzelnen *MountingLink*s (d.h. die Halterungen) eingesammelt und deren jeweiliges *Borehole* ausgelesen. Dieses *Borehole* ist an derselben Stelle, an der in der Entwurfssprache für Taschen ein *Bohrloch* platziert werden soll. Daher werden die Koordinaten des *Borehole* im Koordinatensystem der *Plattenseite* berechnet. Dies geschieht mithilfe der bereits erwähnten Geometriesprache [65, ab Seite 84f]. Für jedes *Borehole* der *MountedBox* wird für die *Plattenseite* ein *Bohrloch* im lokalen Koordinatensystem erstellt. In Abb. 4.78 ist dies die Transformation von a) nach b). Aus den *Bohrlöchern* der *Plattenseite* werden dann (von b) nach c)) mittels Javacode nach bestimmten Regeln Stege erstellt. Jedes *Bohrloch* liegt dabei auf mindestens einem Steg. Aus den Stegen werden dann die *Tasche*n bestimmt und instanziiert.

Die einzelnen *Tasche*n Instanzen werden auf ihre Größe überprüft und (von d) nach e)) in kleinere *Tasche*n zerteilt, wenn sie größer sind als der Wert `maximaleTaschenflaeche` der *Plattenseite*. Die geometrische Repräsentation der *Tasche*n wird nach diesem Ablauf mit einer graphischen Regel aufgebaut.

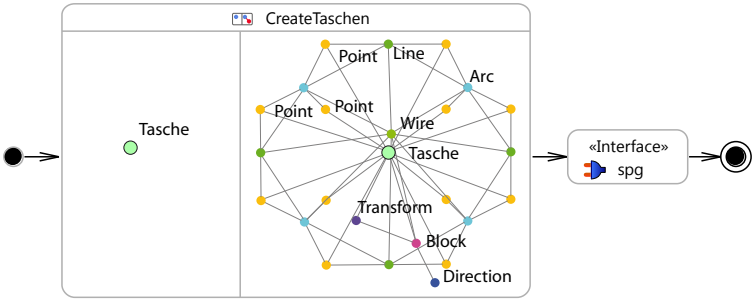


Abbildung 4.79: Erstellen der Taschengeometrie

In Abb. 4.79 ist der Ablauf mit der graphischen Regel für den Taschenaufbau dargestellt. Auf der rechten Seite der Regel kann man die Topologie der *Tasche*ngeometrie erkennen. Die Tasche besteht aus Punkten, die an den Ecken Kreisbögen (*Arc*) definieren. Diese Kreisbögen werden über Linien (*Line*) miteinander verbunden. Daraus ergibt sich eine Kontur (*Wire*), die mit einer Richtungsangabe (*Direction*) zu einem *Block* extrudiert wird. Dieser *Block* wird dann mit einer booleschen Operation von der Platte abgezogen.

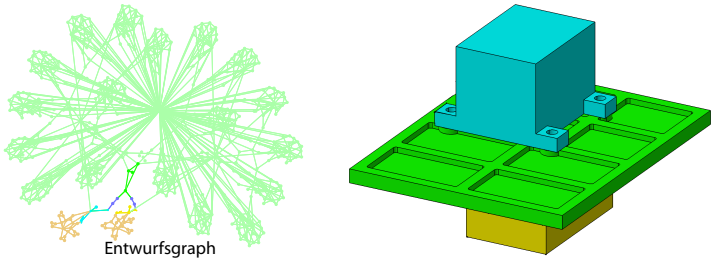


Abbildung 4.80: Entwurfsgraph und Geometrie mit Taschen

In Abb. 4.80 ist das Ergebnis des Aufrufs der Entwurfssprache für Taschen dargestellt. Man kann erkennen, dass sich alle Halterungen auf Stegen befinden. Die Stege sind an den Stellen, wo die Schrauben eingefügt werden, etwas verstärkt durch eine runde Ausbuchtung. Nach dem Erstellen der Taschen kann mit der Verkabelung des Satelliten begonnen werden. Dies geschieht mit einer von Marc Eheim [17] entwickelten Entwurfssprache für Routing.

4.4.5 Routing

In Abb. 4.81 ist ein Ausschnitt aus dem Klassendiagramm der Entwurfssprache für Routing [17] abgebildet. Die für dieses Beispiel wichtigen Klassen sind darin enthalten. Für die Verkabelung

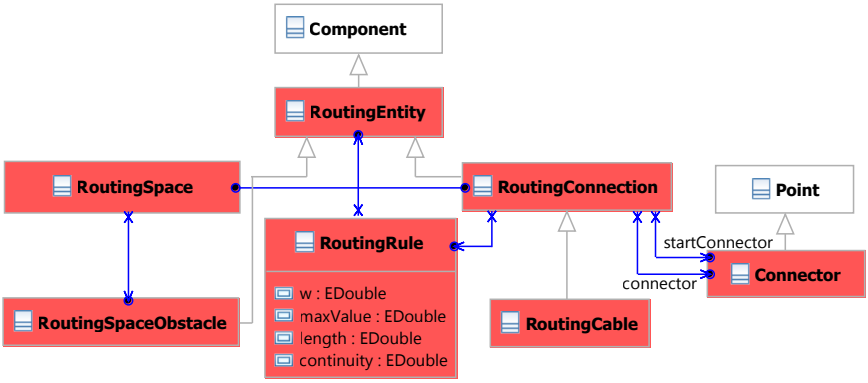


Abbildung 4.81: Klassen für Routing [17]

muss zunächst ein *RoutingSpace* definiert werden. Dies ist der Bauraum in dem der Algorithmus nach möglichen Routen sucht. Alle Bauteile des Satelliten werden als *RoutingSpaceObstacle*s markiert und stellen damit Hindernisse für das Routing dar. Die Verbindungen werden als *RoutingConnection* zu dem *RoutingSpace* hinzugefügt. Um den Algorithmus zu beeinflussen, können *RoutingRule*s definiert werden. Diese setzten beliebige *RoutingEntities* zueinander in Beziehung. So können sich zum Beispiel Kabel gleichen Typs (z.B. Datenkabel) anziehen oder Bauteile bestimmte Kabel anziehen oder abstoßen. Eine Verbindung zwischen zwei Bauteilen wird als *RoutingConnection* zwischen den an den Bauteilen angebrachten *Connector*en modelliert. Ein *RoutingCable* ist die hier benutzte spezielle Form einer *RoutingConnection*. Für eine umfängliche Beschreibung der Routingsprache sei auf die in Entstehung befindliche Dissertation von Marc Eheim [17] verwiesen.

In Abb. 4.82 ist dargestellt, dass die drei *Component*s *Transmitter*, *MidPanel* und *PCDU* alle *RoutingSpaceObstacle*s sind und damit Hindernisse im Bauraum darstellen.

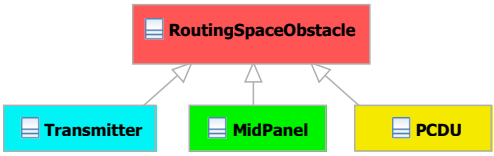


Abbildung 4.82: Kennzeichnen der Hindernisse für das Routing

Um den `startConnector` und den (Ziel-)`connector` für die *RoutingConnection* zwischen den beiden Elektronikboxen festzulegen, werden an den Elektronikboxen zunächst Steckerhüllen angebracht. Die geschieht in der Aktivität in Abb. 4.83. In den ersten beiden

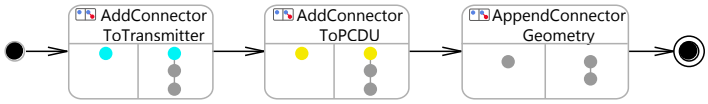


Abbildung 4.83: Einbau der Steckerhüllen

Regeln wird jeweils die *Component* der Elektronikbox gesucht und eine *Position* mit einer *Component* eines bestimmten Steckertyps angehängt. Die letzte Regel wird dann auf beiden der angehängten Steckerkomponenten ausgeführt und fügt eine vordefinierte STEP-Geometrie unter der *Component* ein. In Abb. 4.84 ist auf der linken Seite die Erweiterung des Entwurfsgraphen durch die Geometrie der Steckerhüllen zu sehen. Auf der rechten Seite sind die

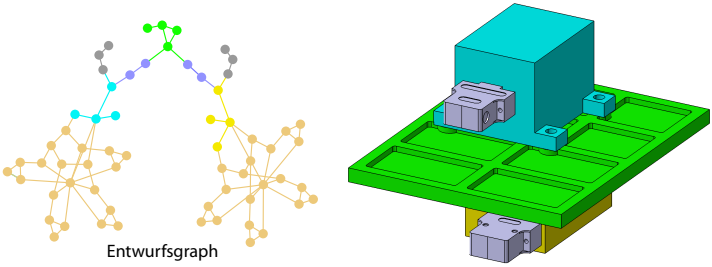


Abbildung 4.84: Geometrie und Entwurfsgraph mit Steckerhüllen

beiden Boxen mit angebrachten Steckerhüllen abgebildet. Die Steckerhüllen werden im lokalen Koordinatensystem der Boxen positioniert und verschieben sich daher mit den Boxen.

In Abb. 4.85 ist die Aktivität zur Verkabelung des Beispiels abgebildet. Es wird zunächst ein

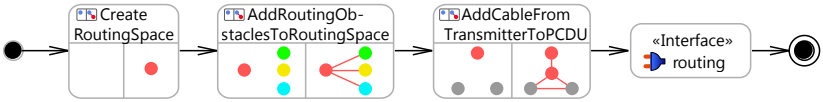


Abbildung 4.85: Aktivität zur Verkabelung

RoutingSpace erstellt. In diesem Beispiel ist das ein einfacher Quader (*Cuboid*). Dann werden alle *RoutingSpaceObstacle*s mit dem *RoutingSpace* verbunden. Erst durch diese Verbindung werden sie beim Routing berücksichtigt. In der dritten Regel „AddCableFromTransmitterToPCDU“ wird das *RoutingCable* zwischen den beiden *Connector*en der Elektronikboxen

eingebaut. Die *Connector*en wurden mit den Steckerhüllen bereits eingebaut. Danach wird der Routing-Algorithmus aufgerufen.

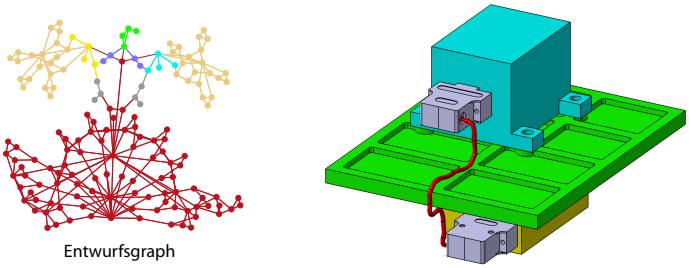


Abbildung 4.86: Entwurfsgraph und Geometrie mit allen Verfeinerungen

In Abb. 4.86 ist die Geometrie nach Ausführen der Routingsprache abgebildet. Das Kabel zwischen den beiden Boxen wird mittels eines Kreises, der an einem Spline extrudiert wird, dargestellt. Im Entwurfsgraph auf der linken Seite ist die detaillierte Abbildung des Kabels durch die roten Knoten im Graph zu erkennen.

4.5 FireSat Integration

In diesem Abschnitt wird die Integration mit Entwurfssprachen anhand des FireSat-Beispiels erläutert. Dafür wird die Synthese detaillierter Simulationsmodelle für drei Beispielvarianten des Satelliten automatisiert durchgeführt. Nach der zuvor vorgestellten analytischen Auslegung des Systems folgt in der Aktivität in Abb. 4.87 die geometrische Konfiguration und Integration des Satelliten.

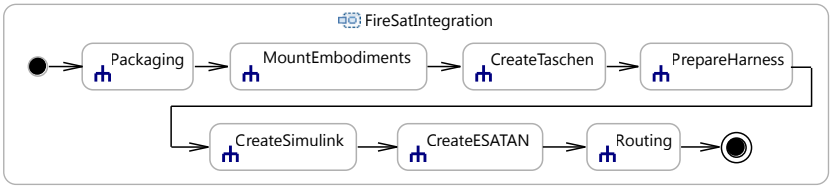


Abbildung 4.87: Automatische Modellsynthese des FireSat-Beispiels

In der Aktivität in Abb. 4.87 wird als erstes in der Subaktivität *Packaging* die Anordnung der Komponenten vorgenommen. Für die nachfolgenden Beispiele wird die Anordnung durch manuell definierte Regeln festgelegt, da der Anordnungsalgorithmus bei den dabei auftretenden dichten Packungen noch nicht vollständig funktioniert. Da in diesem Beispiel die Komponenten durch Regeln platziert werden, kann das *Packaging* für die nachfolgende Bilderserie als erstes

erfolgen. Wird die automatische Anordnung verwendet, so werden zuerst die Steckerhüllen und die Halterungen an die Boxen angebaut, damit sie vom Algorithmus berücksichtigt werden. In der Subaktivität *MountEmbodiments* werden die Komponenten mit der Halterungssprache auf die verschiedenen Einbauebenen des Satelliten montiert. Die Subaktivität *CreateTaschen* ruft die Entwurfssprache zur Erstellung der Versteifungstaschen auf. In der Subaktivität *PrepareHarness* werden die Komponenten für die Verkabelung vorbereitet. Dabei werden unter anderem die Steckerhüllen an den Komponenten platziert. Das Geometriemodell kann zu jedem Zeitpunkt im Entwurfsablauf angezeigt werden. In den folgenden Abbildungen 4.88 bis 4.91 wird das Geometriemodell nach jeder Subaktivität der ersten Zeile dargestellt.

In der zweiten Zeile der Aktivität in Abb. 4.87 werden die detaillierten Simulationsmodelle aufgebaut. Zunächst wird in *CreateSimulink* ein Simulink-Modell für die Simulation der Lageregelung des Satelliten generiert. Dann wird in der Subaktivität *CreateESATAN* ein vollständiges Thermalsimulationsmodell in ESATAN erstellt. In der letzten Subaktivität *Routing* wird die automatische Verkabelung des Satelliten auf Basis des detaillierten Geometriemodells gestartet.

Alle gezeigten Modelle sind aus der FireSat-Entwurfssprache generiert. Die FireSat-Entwurfssprache verwendet dafür etwa 300 Klassen aus 13 verschiedenen Entwurfssprachen. Die Entwurfssprachen enthalten zusammen einen Regelfundus von 450 graphischen und 26 Java-Regeln. Die Regeln sind in 140 Aktivitäten aufgeteilt. Für die unten dargestellte Variante 1 des FireSat werden 214 graphische und 21 Java-Regeln in 62 verschiedenen Aktivitäten ausgeführt. Während des Entwurfsprozesses wird das Gleichungssystem aus bis zu 5312 Gleichungen 24-mal gelöst. Der Entwurfsprozess dauert 3 Minuten 11 Sekunden, das Generieren des Geometriemodells dauert in OpenCascade 15 Sekunden, in CATIA V5 6 Minuten 46 Sekunden. Das Simulink Modell wird in 58 Sekunden erstellt, das ESATAN-Modell in 10 Sekunden. Die Simulation in Simulink dauert ca. 15 Minuten und die Simulation in ESATAN ca. 2-3 Stunden.

4.5.1 Geometriemodell

Das Geometriemodell des FireSat wird aus den Komponenten der einzelnen Subsysteme aufgebaut. Die Geometrie der Komponenten ist in den Entwurfssprachen der Subsysteme modelliert und die Komponenten werden durch die Berechnung des Gesamtentwurfs automatisch ausgelegt. Die meisten Komponenten des Satelliten sind als quaderförmige Boxen modelliert. In Abb. 4.88 ist das Geometriemodell der Variante 1 des FireSat noch ohne Befestigungslaschen dargestellt. Die Variante 1 entspricht dem Auslegungspunkt bei 30 m Bodenauflösung.

In dem vergrößerten Ausschnitt in Abb. 4.88 kann man den Transmitter für die Telemetriedaten sehen. Die Box hat noch keine Halterungslaschen und die Platte, auf der die Box montiert ist, hat auch keine Taschen. Als erstes werden die Halterungslaschen im Geometriemodell hinzugefügt. In Abb. 4.89 ist das resultierende Geometriemodell dargestellt.

Man kann erkennen, dass der Transmitter nun mit Halterungslaschen versehen ist. Auch die anderen Boxen des Satelliten werden um eine Befestigung mit Halterungslaschen erweitert. Die Positionen der Bohrlöcher in den Halterungslaschen werden dann von der Entwurfssprache für Versteifungstaschen verwendet, um die Positionen der Stege festzulegen. Dies geschieht

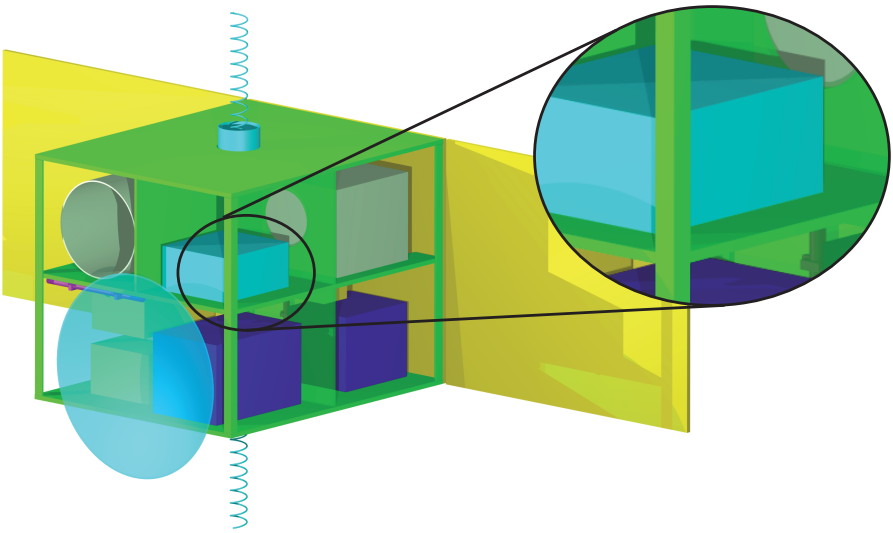


Abbildung 4.88: Variante 1 des FireSat nach „Packaging“

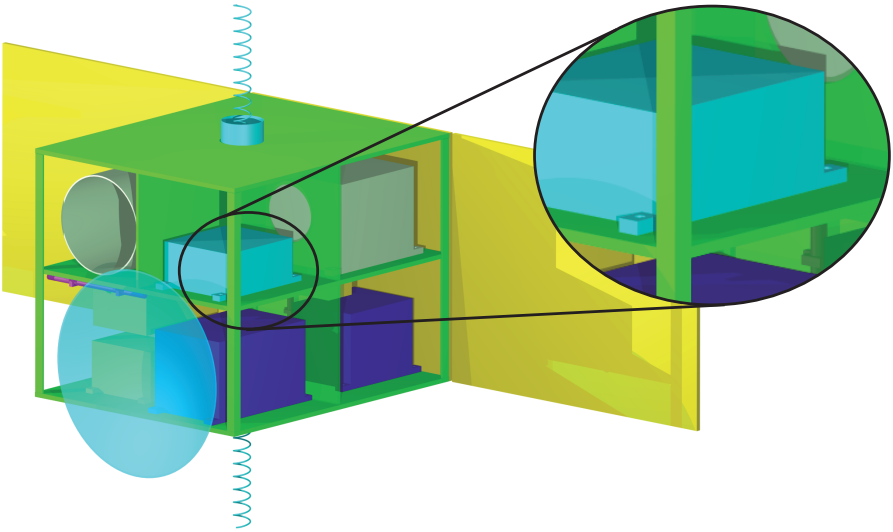


Abbildung 4.89: Variante 1 des FireSat nach „MountEmbodiments“

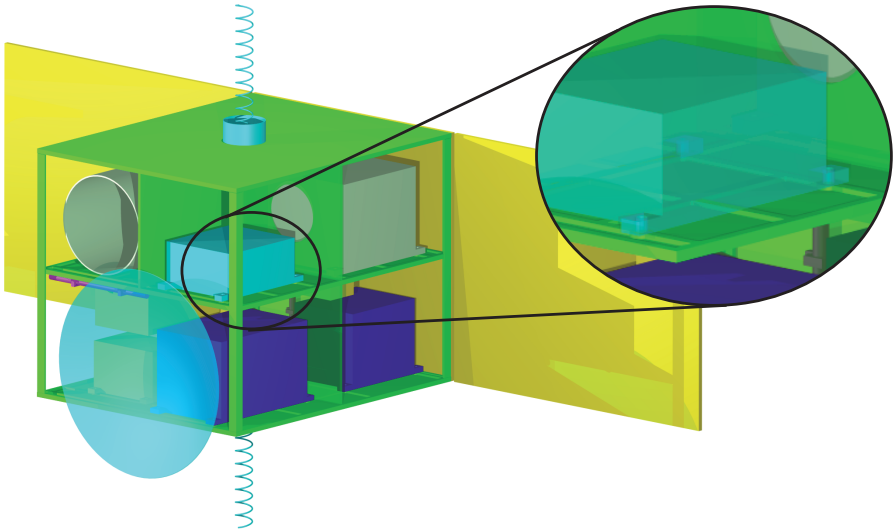


Abbildung 4.90: Variante 1 des FireSat nach „CreateTaschen“

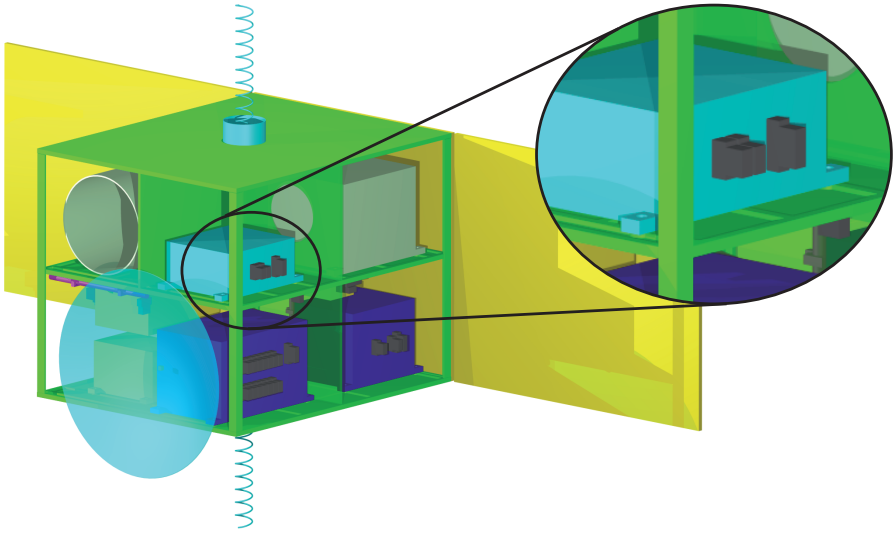


Abbildung 4.91: Variante 1 des FireSat nach „PrepareHarness“

nach dem in Abb. 4.78 dargestellten Algorithmus. Das Ergebnis für das Anbringen der Taschen ist in Abb. 4.90 dargestellt.

In Abb.4.90 ist der Transmitter in dem vergrößerten Ausschnitt transparent dargestellt. Man kann erkennen, dass jeweils an den Enden mit den Halterungslaschen ein Steg quer verläuft. Dieser Steg weist an den Bohrlöchern eine kreisförmige Ausweitung auf, damit genug Material für die Bohrung vorhanden ist. Nach dem Anbringen der Taschen wird das Geometriemodell für die Verkabelung vorbereitet. Dafür wird der Schaltplan für die Verkabelung aufgebaut und die einzelnen Verbindungen festgelegt. Im Geometriemodell ist das Anbringen der Steckerhüllen als Ergebnis dieses Arbeitsschritts sichtbar. In Abb. 4.91 ist das Geometriemodell mit allen Steckerhüllen dargestellt. Damit ist das Geometriemodell fertig erstellt.

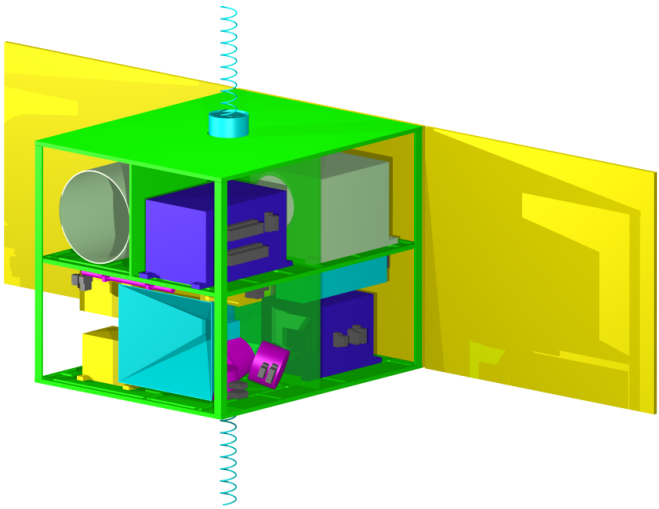


Abbildung 4.92: Variante 2 des FireSat mit Hornantenne

Ändert man im selben Auslegungspunkt (30 m Bodenauflösung) das Lösungsprinzip für die Antenne des Nutzlast-Downlink auf eine Hornantenne, so ergibt sich eine andere Anordnung der Komponenten und damit auch ein anderes Taschenmuster. In Abb. 4.92 ist die Variante 2 des FireSat dargestellt.

Man kann erkennen, dass die Anordnung der Komponenten verschieden ist zu Variante 1. Die Hornantenne kann in den Satelliten integriert werden und macht ihn damit in den äußeren Abmaßen etwas kleiner. Der On-Board Computer ist auf der oberen Ebene untergebracht und die Drallräder sind hinter der Antenne montiert. Zusätzlich zum Transmitter findet rechts hinten eine Wanderfeldröhre Platz. Die Komponenten sind ansonsten zum Großteil gleich wie bei Variante 1, da sich der Auslegungspunkt nicht geändert hat.

Um die Auswirkungen des Auslegungspunktes der Nutzlast auf den gesamten Satelliten in

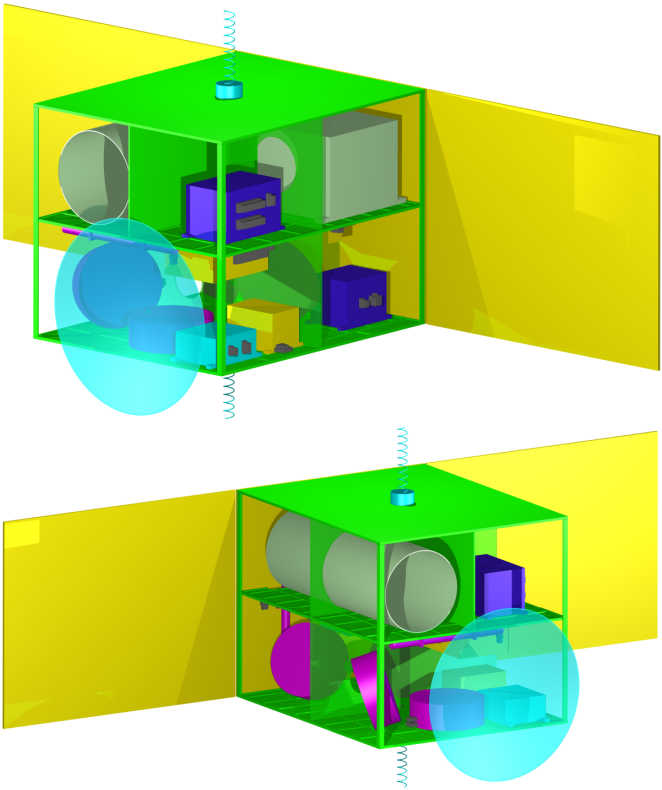


Abbildung 4.93: Variante 3 des FireSat mit 20m Auflösung

der Integration zu untersuchen, wird die Variante 3 des FireSat ausgelegt. Die Variante 3 ist mit der Anforderung einer Bodenauflösung von 20 m entworfen. Diese Anforderung hat eine starke Auswirkung auf die Größe der Nutzlast und, wie in Abb. 4.54 bereits gezeigt, auch auf die Masse des Gesamtsystems. In Abb. 4.93 ist das Geometriemodell der Variante 3 dargestellt.

Auffällig bei dieser Variante sind die Größe der Struktur und die großen Drallräder. Durch die Anforderung an die Auflösung wird das Teleskop sehr viel länger und da die Struktur aufgrund des Teleskops ausgelegt wird, erscheint der Satellit in dieser Variante sehr geräumig. Die Drallräder ergeben sich aus dem größeren Trägheitsmoment des Satelliten und auch die Magnettorquer sind größer.

4.5.2 Lageregelungssimulation

In der Entwurfssprache für das Lageregelungssystem wird, wie in Abschnitt 4.2.3 bereits gezeigt, für die Simulation ein Aktivitätsdiagramm aufgebaut. Dieses Aktivitätsdiagramm beschreibt die Regelung des Satelliten und kann nach Matlab Simulink exportiert werden. In Abb. 4.94 ist die oberste Ebene des exportierten Simulink-Modells abgebildet.

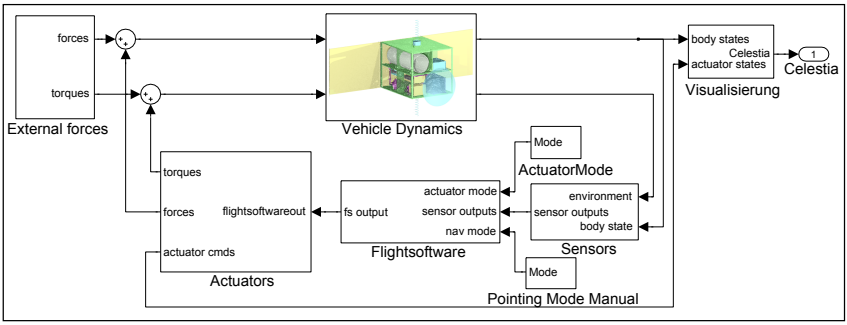


Abbildung 4.94: Generiertes Simulink-Modell

Das Simulink-Modell in Abb. 4.94 entspricht dem Aktivitätsdiagramm in Abb. 4.34. Die Bewegung des Satelliten wird mit einer Starrkörperbewegung in dem Block „VehicleDynamics“ berechnet. Die Lage (Winkel) und Drehung (Winkelgeschwindigkeiten) des Satelliten werden an den Block „Sensors“ weitergegeben und dort durch die Modelle der Sensoren umgewandelt in Messwerte. Diese werden an den Block „Flightsoftware“ zur Berechnung der Aktuatorkommandos übertragen. Im Block „Flightsoftware“ werden die Kommandos für die Aktuatoren durch ein PD-Reglermodell berechnet. Die Modelle der Aktuatoren im Block „Actuators“ werden durch die berechneten Kommandos gesteuert und geben daraufhin die Kräfte auf den Satelliten zurück. Danach werden die externen Kräfte (solarer Strahlungsdruck, Restatmosphäre etc.) zu den Steuerkräften addiert und der Kreislauf beginnt von neuem.

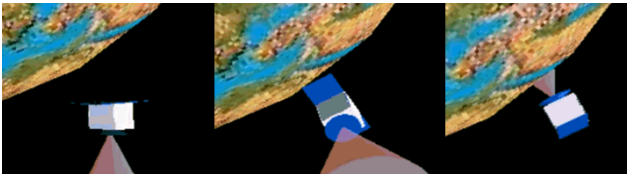


Abbildung 4.95: FireSat-Ausrichtungsvorgang

In Abb. 4.95 ist die Visualisierung in einem VRML-Modell dargestellt. Der FireSat in der Sequenz in Abb. 4.95 wechselt vom ungeregelten Modus in einen Nadir-Pointing Mo-

dus. Der Sendestrahl der Antenne ist als Konus dargestellt. Dieses Manöver soll nun für die drei verschiedenen Varianten des FireSat genauer untersucht werden. In Abb. 4.96 ist das Simulationsergebnis der Variante 1 des FireSat für das Manöver dargestellt.

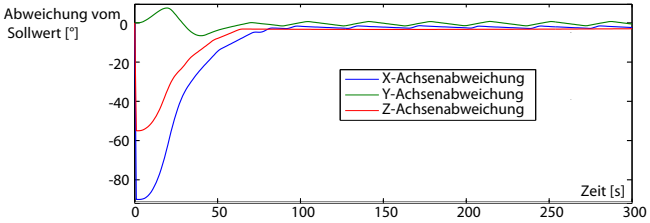


Abbildung 4.96: Simulation des Ausrichtungsvorgangs der Variante 1

Die drei Linien in Abb. 4.96 stellen jeweils die Abweichung des Winkels einer Achse zum Sollwert dar. Zu Beginn der Simulation wird der Nadir-Pointing Modus aktiviert. Die Abweichungen der drei Achsen vom Sollwert sind auf der Y-Achse des Diagramms in Grad dargestellt. Auf der X-Achse ist die simulierte Zeit in Sekunden aufgetragen. Der Satellit regelt die drei gemessenen Winkel auf den Sollwert hin. Man sieht, dass die Regler zum Teil mehrere Überschwinger erzeugen und erst nach einer gewissen Zeit den Sollwert in allen drei Achsen erreichen. Diese Variante des Satelliten erreicht den Sollwert innerhalb von ca. 70 Sekunden. Danach bewegen sich die drei Achsen innerhalb der Genauigkeitsgrenzen des Reglers.

Die Zeit bis der Sollwert erreicht ist sagt aus, wie schnell der Satellit durch ein Kommando auf eine bestimmte Ausrichtung gebracht werden kann. Für den FireSat ist dies von Bedeutung, wenn auf Kommando ein bestimmtes Gebiet auf der Erde beobachtet werden soll. Durch die Simulation der Lageregelung wird die funktionale Anforderung an die Ausrichtzeit überprüft.

Die Variante 2 des FireSat hat in etwa die gleiche Masse und die gleichen Trägheitsmomente sowie ähnliche Aktuatoren wie Variante 1. Die Simulation ergibt in etwa die gleichen Werte. Interessant ist hingegen die in Abb. 4.97 dargestellte Simulation der Variante 3 des FireSat. Dieser wesentlich größere Satellit hat ein größeres Trägheitsmoment und größere Aktuatoren.

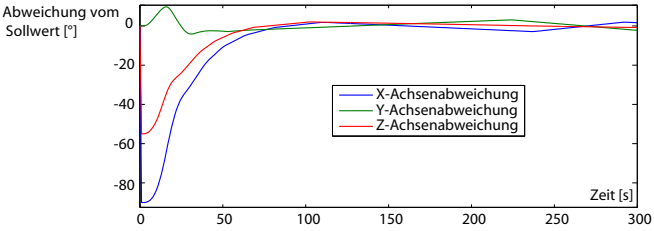


Abbildung 4.97: Simulation des Ausrichtungsvorgangs der Variante 3

In der Simulation in Abb. 4.97 kann man erkennen, dass der Satellit ebenfalls eine sehr rasche Drehbewegung ausführt und die Lage innerhalb von ca. 75 Sekunden erreicht. Eine Regel legt anhand der zu erreichenden Drehgeschwindigkeit die Größe der Drallräder fest. Da ein größerer Satellit durch den höheren Steiner-Anteil im Trägheitsmoment jedoch schwieriger zu beeinflussen ist, führt die Regel zum Einbau relativ großer Drallräder. In Abb. 4.98 sind die Drallräder auf der linken Seite in lila dargestellt. Um den Satelliten etwas leichter zu machen, wird die geforderte Drehgeschwindigkeit in der Entscheidungsregel reduziert. Dies führt zu dem in Abb. 4.98 auf der rechten Seite dargestellten Satelliten.

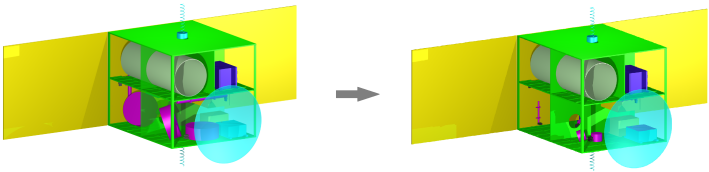


Abbildung 4.98: Geometrie Variante 3 vor und nach Regeländerung

Die Simulation der Aktivierung des Nadir-Pointing Modus mit der veränderten Variante 3 des FireSat ist in Abb. 4.99 dargestellt. Man kann sehen, dass das Manöver mit den kleineren Aktuatoren wesentlich länger dauert. Die Ausrichtung des Satelliten ist erst nach ca. 150 Sekunden erreicht. Demgegenüber steht jedoch ein um 29,5 kg geringeres Systemgewicht.

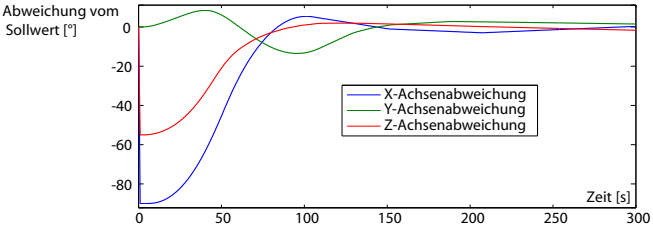


Abbildung 4.99: Ausrichtungsvorgang der Variante 3 nach Regeländerung

Durch die oben dargestellten Simulationen soll verdeutlicht werden, wie es in einer Entwurfssprache möglich ist, die Auswirkung von operationellen Anforderungen einer Mission auf die Hardware des Satelliten-Systems zu untersuchen. Des weiteren können „trade-off“ Studien durchgeführt werden um das für die Mission angepasste Optimum zwischen der Leistungsfähigkeit des Systems und den Kosten (hier in Form von Masse) zu finden.

4.5.3 Thermalsimulation

Eine der wichtigsten Simulationen in der Raumfahrt stellt die Thermalsimulation dar. In der Thermalsimulation wird die Entstehung und Verteilung von Wärmeenergie innerhalb des Systems sowie die Aufnahme und Abgabe von Strahlungsenergie von und zur Umwelt untersucht. Für die Thermalsimulation wird das System in Thermalknoten diskretisiert. Die Thermalknoten repräsentieren die thermalen Eigenschaften der Komponenten und sind untereinander verbunden. Die Verbindungen geben die Kopplung zwischen den Knoten an, wodurch die Wärmeströme berechnet werden können.

Die Simulation wird in dieser Arbeit in der Berechnungsumgebung ESATAN-TMS durchgeführt. In einer vom Autor betreuten Diplomarbeit [32] wurden die Grundlagen für eine Entwurfssprache für die Thermalsimulation entwickelt. In Abb. 4.100 sind die Klassen für die Geometriemodellierung im Thermalmodell dargestellt.

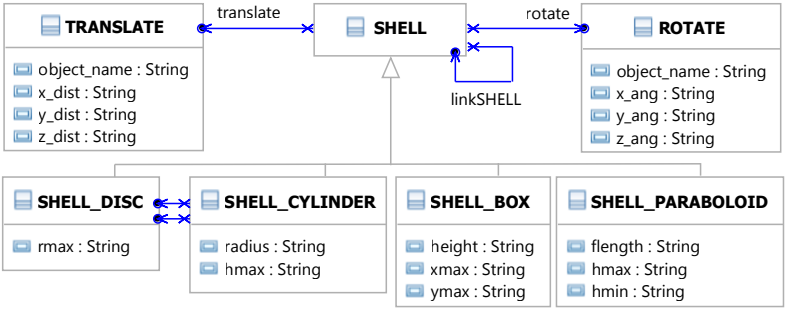


Abbildung 4.100: Klassen für Thermalgeometrie

Das Geometriemodell besteht in der Thermalsimulation aus Oberflächen. Die Klasse *SHELL* ist eine Oberklasse für alle geometrischen Oberflächenelemente. Eine *SHELL* kann weitere *SHELL*s enthalten und wird in diesem Fall zu einer Baugruppe. Eine *SHELL* kann man im Raum verschieben (*TRANSLATE*) und rotieren (*ROTATE*). Im unteren Teil von Abb. 4.100 sind einige Primitive der Entwurfssprache abgebildet. Die Primitive erben von *SHELL* und besitzen verschiedene Attribute. Eine *SHELL_BOX* hat zum Beispiel eine `height` für die Höhe und einen Wert `xmax` und `ymax` für die Länge und die Breite.

Der Wärmeaustausch in einer Thermalsimulation für Satelliten basiert auf den Prinzipien der Strahlung und der Wärmeleitung. In Abb. 4.101 sind die relevanten Klassen aus der Entwurfssprache abgebildet. Mit einer *SIDE1* und *SIDE2* werden die Anfangsknotennummern (`nbase`) und die optischen Eigenschaften (`opt`) der beiden Oberflächen beschrieben. Das *MESH* definiert die Anzahl der Knoten (`nodes`) in die beiden Richtungen einer Oberfläche und das *BULK_MATERIAL* definiert die angenommene Dicke (`thick`) des Materials (`bulk`). Die Wärmekapazität eines Körpers leitet sich dann aus der Dicke mal der Oberfläche der in dem Material vorgegebenen Dichte und der spezifischen Wärmekapazität ab. Um die Kontaktwärme-

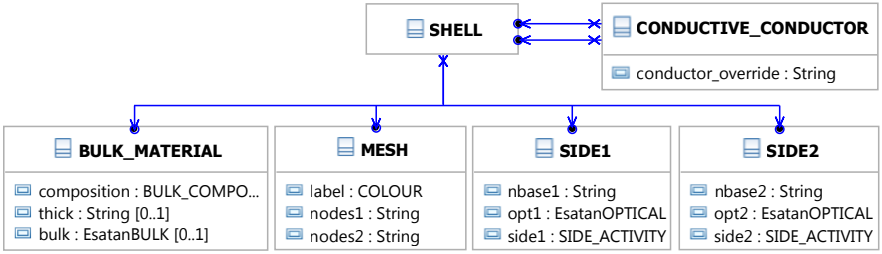


Abbildung 4.101: Klassen für Thermalmodellierung

leitung zu modellieren gibt es einen *CONDUCTIVE_CONDUCTOR*, der einen Wärmestrom (`conductor_override`) in der Einheit W/K definiert.

Um das Thermalmodell für dieses Beispiel zu erstellen, wird die Aktivität *CreateESATAN* in Abb. 4.102 ausgeführt. Für die Erstellung der Thermalsimulation wird zunächst die Geo-

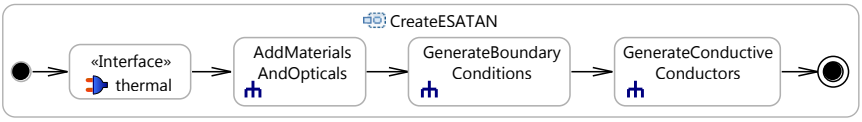


Abbildung 4.102: Aufbau des FireSat-Thermalmodells

metriepräsentation mit den Netzen über einen Interfaceaufruf erstellt. Bei der Erstellung der Thermalgeometrie werden nur die *Shapes* berücksichtigt, die über einen `primitiveShape`-Link mit einer *Component* verbunden sind. Aus diesem Grund wirken sich zum Beispiel die Transformationen zum Anbringen der *MountingLink*s nicht auf die Geometrie im Thermalmodell aus. Als Ergebnis des Exports ist in Abb. 4.103 das Geometriemodell der Variante 1 des FireSat abgebildet.

Man kann erkennen, dass die Komponenten des Satelliten als Boxen, Zylinder und andere Primitive dargestellt werden. Die Geometrie muss für die Simulation einfach gehalten werden, damit die Anzahl der Thermalknoten nicht zu groß wird. Daher sind die Details wie zum Beispiel die Halterungslaschen und die Taschen nicht in dem Modell abgebildet.

Wenn die Geometrie erstellt ist, können die Materialien und Oberflächeneigenschaften den Komponenten zugewiesen werden. Dies erfolgt über eine einfache Liste, die den Komponenten nach Namen das zugehörige Material und die Opticals zuordnet. Danach werden die Randbedingungen definiert. Alle *PowerElements* verbrauchen Strom, der in der Eigenschaft `power` angegeben ist. Aus diesem Zusammenhang wird für die Thermalsimulation eine Randbedingung erstellt, dass die Komponenten diesen Strom als Wärmeleistung abgeben.

Im letzten Schritt werden dann die *CONDUCTIVE_CONDUCTORS* erstellt. Dies wird aufbauend auf der Halterungssprache vorgenommen. Jede Komponente, die auf einer Einbau-

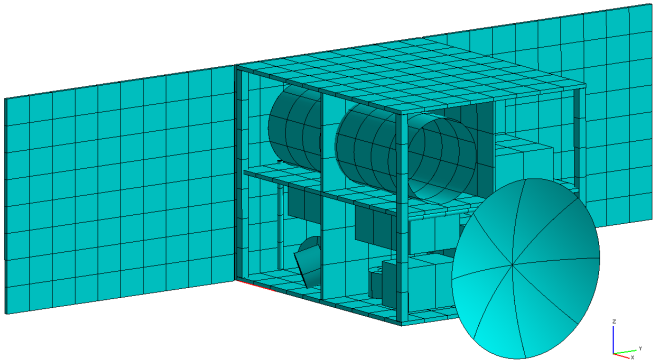


Abbildung 4.103: Geometriemodell Variante 1 in ESATAN-TMS

ebene montiert ist, wird durch einen *CONDUCTIVE_CONDUCTORS* auch thermal an diese gekoppelt. Anschließend wird der gesamte Thermal-Entwurfsgraph in das Thermalprogramm ESATAN-TMS exportiert.

Für die Simulation des Thermalhaushalts kann das Geometriemodell dann noch weiter verfeinert werden. Außen an der Struktur können Radiatoren angebracht werden und Komponenten können mit sogenannter Multi-Layer-Insulation (MLI) eingepackt werden. Dies kann entweder in der Entwurfssprache regelbasiert gemacht werden oder im Anwendungsprogramm durch eine manuelle Bearbeitung erfolgen. In diesem Beispiel wird nur eine der drei Varianten verfeinert, daher wird die manuelle Bearbeitung gewählt. In Abb. 4.104 ist das verfeinerte Geometriemodell mit einem Temperaturplot abgebildet.

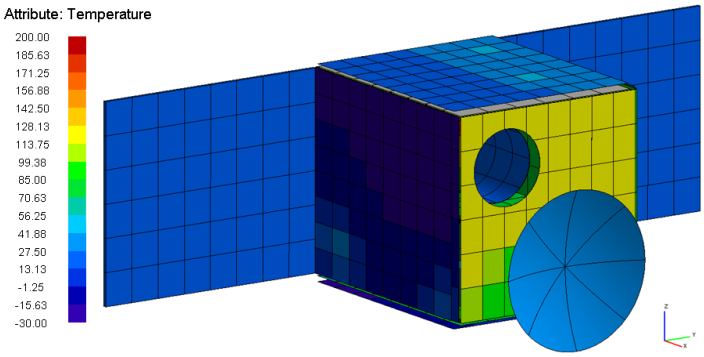


Abbildung 4.104: Simulation der Variante 1

Auf dieser Abbildung ist zu sehen, dass die Außenseiten oben und links noch mit Radiatoren versehen sind. Unter den Radiatoren wird der FireSat mit sogenannter Multi-Layer-Insulation (MLI) eingepackt. Die Vorderseite des Satelliten in Abb. 4.104 ist mit 90-110 Grad Celsius relativ heiß. Die hohen Temperaturen entstehen durch die eingestrahlte Wärmeenergie der Erde. Die MLI erreicht dabei schnell hohe Temperaturen, da sie keine große Wärmekapazität besitzt und hauptsächlich nach außen wieder abstrahlt. In Abb. 4.105 ist das Simulationsergebnis im Inneren des Satelliten unter diesen Randbedingungen dargestellt.

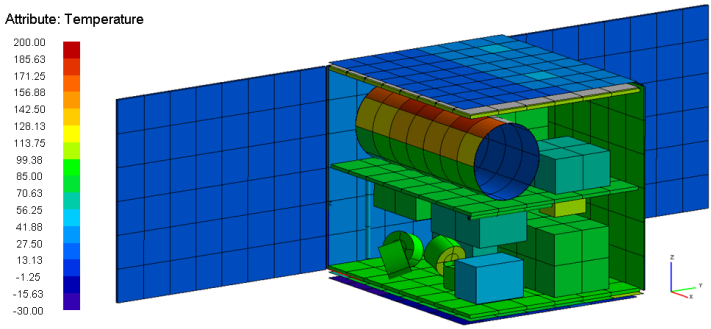


Abbildung 4.105: Temperaturverteilung der Komponenten der Variante 1

In Abb. 4.105 ist die Oberfläche des Teleskops mit Temperaturen um 200 Grad Celsius hervorzuheben. Diese hohen Temperaturen entstehen jedoch auch, weil das Teleskop zum Schutz in MLI eingepackt ist. Im Inneren des Teleskops herrschen hingegen verhältnismäßig niedere Temperaturen. Als heißeste Komponente könnte der *PowerConverter* im vorderen rechten Bereich problematisch werden. Der *PowerConverter* wird so heiß, da er die gesamte von den *SolarPanels* erzeugte Energie umwandeln muss.

Neben den funktionalen Eigenschaften des Entwurfs wird für die Simulation auch die Definition des Orbits und der Ausrichtung des Satelliten benötigt. Der Orbit wird durch die Bahnhöhe und die Inklination definiert. Da es sich um einen Kreisorbit handelt ist keine Exzentrizität angegeben. Diese Informationen werden aus den Instanzen der Systembibliothek ausgelesen und als Randbedingungen für die Simulation übergeben. In Abb. 4.106 ist der FireSat im Nadir-Pointing Modus dargestellt.

Um die Simulation des Energiebudgets des Satelliten innerhalb von ESATAN-TMS zu ermöglichen, wird für die Simulation der produzierten Energie eine FORTRAN Subroutine erstellt. Durch die Sichtfaktoren zur Sonne muss man nur die Thermalknoten der Solarzellen kennen, dann kann man die eingestrahlte Energie bestimmen. Da im Modell die *SolarPanels* bekannt sind, können durch eine einfache Regel alle Knoten der *SolarPanels* ausfindig gemacht und die Knotennummern als FORTRAN-Code ausgegeben werden. Dieses Vorgehen wird in [19] genauer erläutert.

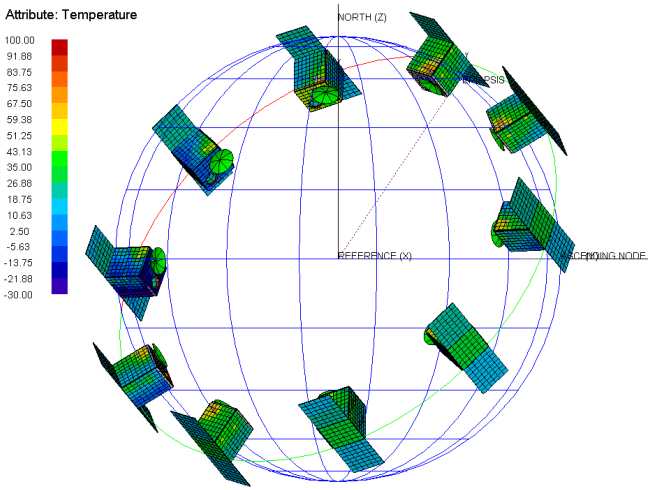


Abbildung 4.106: Temperaturverteilung über einen Orbit

Für die beiden anderen Varianten des FireSat wird hier nur die Generierung der Geometrie gezeigt, um den generischen Aspekt der Modellgenerierung mit Entwurfssprachen zu beleuchten. In Abb. 4.107 ist das Thermalmodell der Variante 2 des FireSat abgebildet. Bei dem Satelliten mit Hornantenne wird für die Übertragung der Hornantenne in das Thermalmodell eine Eigenschaft des Thermalmodellers genutzt. Die Hornantenne wird in der abstrakten Geometriemodellierung über einen Loft definiert. Da es in der Thermalmodellierung in ESATAN

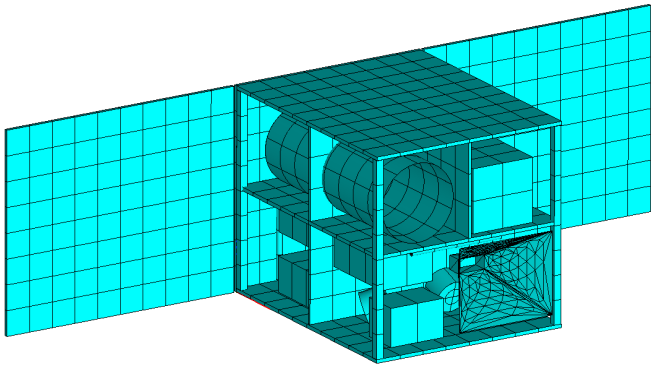


Abbildung 4.107: Geometriemodell Variante 2 in ESATAN-TMS

keine Loft-Operation gibt, wird die Abbildung generisch über ein Netz gelöst. Dabei wird mithilfe des OpenCascade-Kernels ein Geometriemodell erstellt und vernetzt. Die Dreiecke des Netzes werden dann in Triangles in ESATAN übersetzt. Damit ist es nun möglich in einem Programm, das normalerweise nur strukturierte Netze hat, aber ein Dreieckselement besitzt, beliebige Geometrien mit unstrukturierten Gittern zu verwenden.

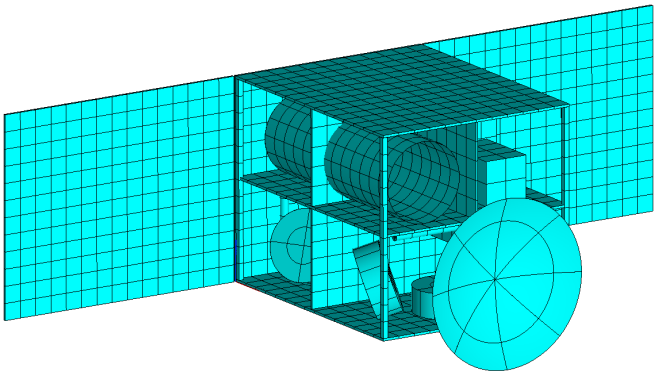


Abbildung 4.108: Geometriemodell Variante 3 in ESATAN-TMS

In Abb. 4.108 ist das Thermalmodell der Variante 3 des FireSat dargestellt. Die Anzahl der Knoten auf den Solarpanelen erscheint hier etwas hoch. Dies liegt an den Abmessungen des Satelliten. Bei der Generierung wird derzeit eine feste Anzahl an Knoten pro Länge vorgegeben. Dadurch werden die großen Strukturen dieser FireSat-Variante scheinbar sehr fein vernetzt, in Wirklichkeit ist die Auflösung jedoch genauso groß wie in Abb. 4.107.

4.5.4 Verkabelung

Die Verkabelung des Satelliten erfolgt automatisch anhand eines Verkabelungs-Algorithmus. Für die Verkabelung werden als erstes in einer für Satelliten angepassten Verkabelungs-Entwurfssprache die Stecker an den Komponenten und die dazugehörigen Kabel definiert. Die Positionen der Stecker an den Komponenten werden von den jeweiligen Subsystemen vorgegeben. Durch Regeln wird dann eine abstrakte Verkabelungstopologie erstellt. Die Regeln definieren zum Beispiel, dass jedes *PowerElement* durch ein Stromkabel mit der *PCDU* verbunden wird. Ebenso wird jedes *DataElement* durch ein Datenkabel mit dem *OBC* verbunden. Diese beiden Regeln legen das Grundgerüst für die Sterntopologie der Daten- und Powerverkabelung fest. Auf diese Weise wird ein großer Teil der Verkabelung durch einfache Regeln definiert. Die speziellen Kabel der Subsysteme werden durch gesonderte Regeln in den Subsystemen selbst definiert. So sind zum Beispiel die Verbindung zwischen dem Teleskop und der Nutzlast-Elektronik in der FireSat-Entwurfssprache und die Verbindungen zwischen den

Transmittern und den dazugehörigen Antennen in der Kommunikationssystem-Entwurfssprache definiert.

Aus der abstrakten Verkabelungstopologie werden die konkreten Kabelverbindungen generiert. Diese Kabelverbindungen werden in einem diskretisierten Bauram als Start- und Zielpunkte für den Routing-Algorithmus verwendet. Die Verkabelung des Satelliten wird dann mit Hilfe der Routing-Entwurfssprache von Marc Eheim [17] durchgeführt. Für die Routing-Entwurfssprache werden Regeln definiert, nach denen der Routing-Algorithmus gesteuert wird. So kann man durch die Regeln erreichen, dass die Verkabelung den Anforderungen im Satellitenbau entspricht. In Abb. 4.109 ist die Verkabelung der Variante 1 des FireSat dargestellt.

Auf den beiden oberen Bildern sind alle Kabel rot dargestellt. Auf den beiden unteren Bildern ist der Power-Kabelbaum rot dargestellt, der Daten-Kabelbaum gelb, die Kabel der Nutzlast grün und die Antennenkabel türkis. Gut zu erkennen sind jeweils die *PCDU* mit einer starken Bündelung der Powerkabel und der *OBC* mit einer starken Bündelung der Datenkabel. Die Kabel enden auf den Bildern jeweils an den Anschlussstellen der Stecker. Diese Anschlussstellen werden durch die Positionen der Stecker auf den Bauteilen definiert. Um

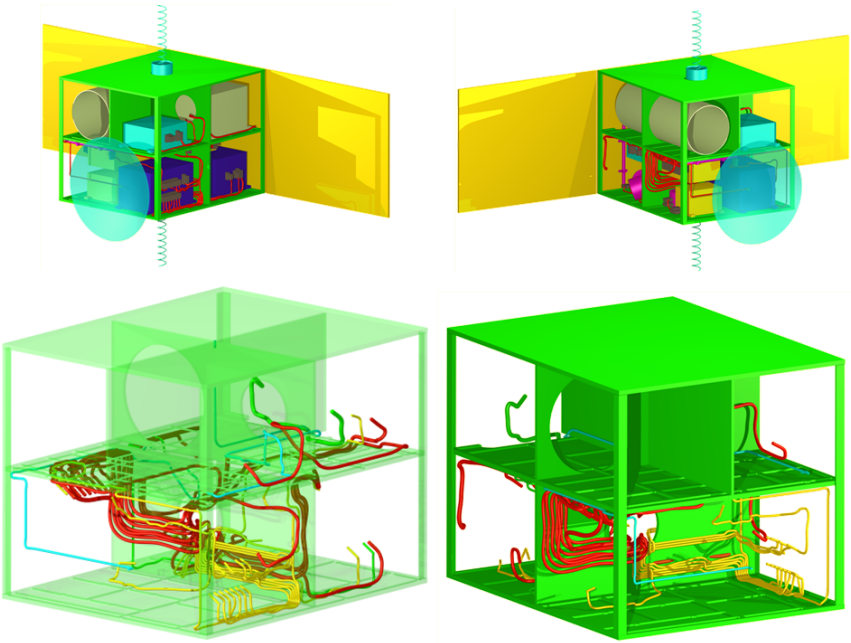


Abbildung 4.109: Verkabelung Variante 1

den Einfluss der Verkabelungsregeln auf den Kabelbaum zu demonstrieren, sind in Abb. 4.110 drei verschiedene Verkabelungen für die Variante 1 des FireSat dargestellt. Die Anordnung der Komponenten ist die gleiche wie in Abb. 4.109.

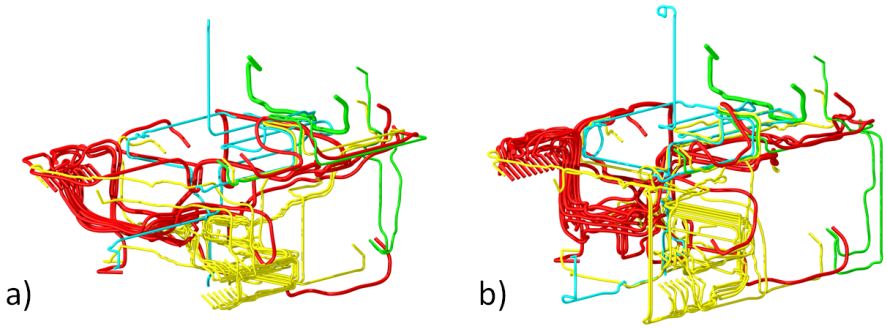


Abbildung 4.110: Verkabelung mit unterschiedlichen Regeln

Für die Steuerung des Verkabelungsalgorithmus werden zwei Arten von Regeln verwendet. Die eine Art von Regeln bewirkt eine Anziehung zwischen zwei *RoutingElement*s. Eine solche Regel wird verwendet, um eine Anziehung zwischen den Datenkabeln und eine Anziehung zwischen den Powerkabeln hervorzurufen. Dadurch bilden sich Kabelbäume für Daten- und Powerkabel aus. Des weiteren wird eine Anziehung der Kabel an die Struktur des Satelliten bewirkt. Die andere Art von Regeln bewirkt eine Abstoßung zwischen zwei *RoutingElement*s. Diese Regeln werden verwendet, um eine Abstoßung zwischen Daten- und Powerkabeln hervorzurufen, wodurch die Kabelbäume für Daten- und Powerkabel getrennt verlaufen.

Die Anziehung an die Struktur wird für das Beispiel in Abb. 4.110 variiert. In dem Routing

	Kabelbaum	Anz. Kabel	Masse m [kg]	Länge l [m]
Var. a)	Daten (gelb)	14	1,14	18,6
	Power (rot)	12	2,64	19,4
	Antenna (türkis)	3	0,24	3,97
	Nutzlast (grün)	3	0,31	1,25
	Summe	32	4,34	43,24
Var. b)	Daten (gelb)	14	1,22	19,95
	Power (rot)	12	3,01	22,12
	Antenna (türkis)	3	0,30	4,83
	Nutzlast (grün)	3	0,32	1,29
	Summe	32	4,85	48,18

Tabelle 4.1: Kabellängen und Massen der beiden Routings in Variante 1

a) auf der linken Seite wird eine schwache Wandanziehung verwendet. In dem Routing b) auf der rechten Seite wird eine starke Wandanziehung verwendet. In Tabelle 4.1 sind die Massen und Längen der Kabel für die beiden Routings enthalten.

In den Spalten in Tabelle 4.1 steht zunächst die Variante aus Abb. 4.110, dann sind die vier Kabelbäume aufgelistet, die Datenkabel in gelb, die Powerkabel in rot, die Antennenkabel in türkis und die Kabel der Nutzlast in grün. In der nächsten Spalte steht die Anzahl der gerouteten Kabel. In den Spalten danach sind jeweils für die Summe der Kabel die Masse und die Länge angegeben. Man sieht, dass die Längen der Kabel bei starker Wandanziehung ansteigen.

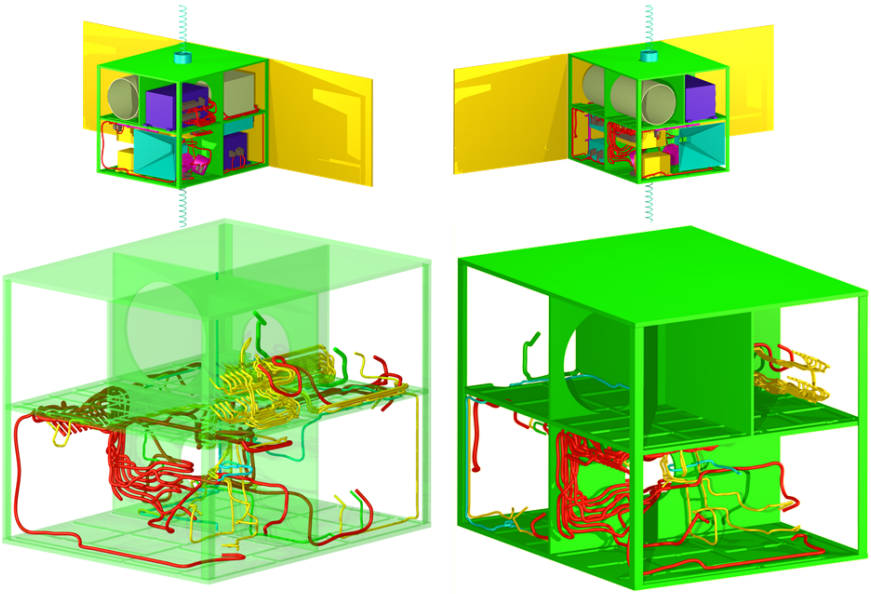


Abbildung 4.111: Verkabelung der Variante 2

Die Verkabelung für die Variante 2 des FireSat ist in Abb.4.111 dargestellt. Man kann erkennen, dass durch die veränderte Position des *OBC* der gesamte Datenkabelbaum von der oberen Ebene des Satelliten ausgeht.

In Abb. 4.112 ist die Variante 3 des FireSat verkabelt. Man sieht, dass durch die deutlich größere Struktur die Kabelbahnen über längere Strecken gerade nebeneinander verlaufen. Auch in dieser Variante ist der *OBC* auf der oberen Ebene untergebracht. Die *PCDU* ist hingegen direkt darunter auf der Unterseite der mittleren Einbauplatte montiert.

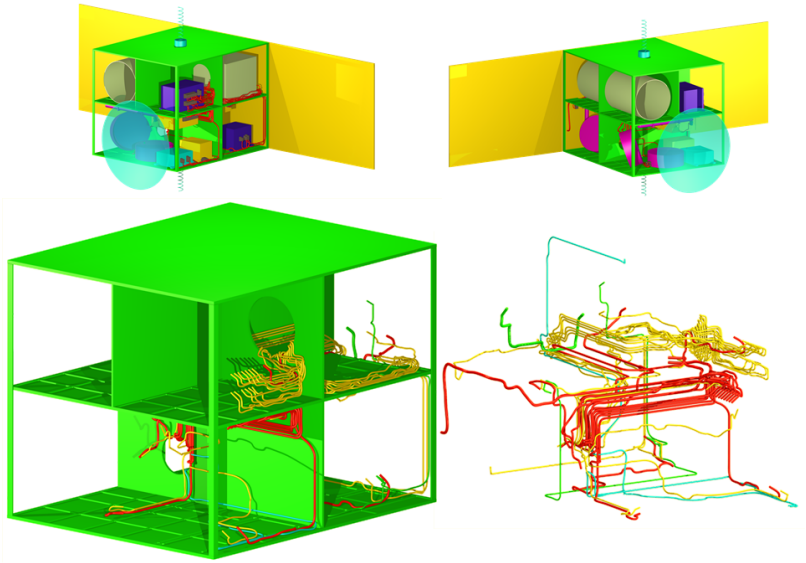


Abbildung 4.112: Verkabelung der Variante 3

Die Daten zu den Verkabelungen der drei Varianten des FireSat sind in Tabelle 4.2 festgehalten. In dieser Tabelle sind die drei verschiedenen Varianten des FireSat übereinander enthalten. Zusätzlich zu Tabelle 4.1 ist in der letzten Spalte die Länge der Kabel einer Kategorie (Farbe) durch die Anzahl der Kabel geteilt. Damit ergibt sich eine mittlere Kabellänge, die eine Aussage über die Kabelwege in der speziellen Konfiguration zulässt. Die mittlere Kabellänge von Variante 2 ist im Daten-Kabelbaum mit 2,08 m wesentlich höher als die mittlere Länge der Datenkabel bei Variante 1b mit 1,43 m. Der Power-Kabelbaum ist bei Variante 2 mit 1,73 m etwas kürzer relativ zu 1,84 m bei Variante 1b. Insgesamt ergibt sich damit bei Variante 2 mit dem *OBC* im oberen Teil des Satelliten ein um 8 m längerer Gesamtkabelbaum. Die Variante 3 hat erwartungsgemäß den längsten Kabelbaum mit knapp 74 m. Dies ist zu erwarten, da die Struktur des Satelliten bei Variante 3 mit 1,26 m deutlich länger ist, als die Varianten 1 und 2 mit 84 cm.

In diesem Abschnitt wurde dargestellt, wie innerhalb einer Entwurfssprache auch kompliziertere Detailfragen, wie zum Beispiel die der Verkabelung automatisiert beantwortet werden können. Dies kann aufgrund der einfachen Integration in den Entwurfsprozess auch schon in einem frühen Entwurfsstadium geschehen. Dadurch kann die Masse des Kabelbaums wesentlich genauer bestimmt werden als durch die bisher üblichen Faustformeln.

	Kabelbaum	Anz. Kabel	Masse m [kg]	Länge l [m]	l/Kabel [m]
Var. 1b	Daten (gelb)	14	1,22	19,95	1,43
	Power (rot)	12	3,01	22,12	1,84
	Antenna (türkis)	3	0,30	4,83	1,61
	Nutzlast (grün)	3	0,32	1,29	0,43
	Summe	32	4,85	48,18	1,51
Var. 2	Daten (gelb)	14	1,78	29,15	2,08
	Power (rot)	13	3,05	22,44	1,73
	Antenna (türkis)	4	0,26	4,34	1,08
	Nutzlast (grün)	2	0,25	0,975	0,49
	Summe	33	5,35	56,89	1,72
Var. 3	Daten (gelb)	13	2,23	36,38	2,8
	Power (rot)	12	4,22	31,06	2,59
	Antenna (türkis)	2	0,30	4,90	2,45
	Nutzlast (grün)	3	0,25	1,65	0,55
	Summe	30	7,17	73,99	2,47

Tabelle 4.2: Kabelmassen und Längen der drei Varianten

5 Zusammenfassung

In der vorliegenden Arbeit wird die Methodik der Entwurfssprachen auf den Satellitenentwurf angewendet. Der Umfang eines Satellitenentwurfs konnte nur durch die konsequente Einschränkung auf die zur Demonstration der Methodik wesentlichen Aspekte bewältigt werden. Die Entwicklung von Entwurfssprachen für die gesamte Breite und Tiefe des gezeigten FireSat-Entwurfs ist ein wesentliches Novum dieser Arbeit.

Die Breite des Beispiels besteht in dem Entwurfsprozess des FireSat, der von den Anforderungen der Mission bis zu den detaillierten Simulationen durchgängig in den Entwurfssprachen abgebildet ist. Die Tiefe des Beispiels zeigt sich an den verschiedenen Stationen des FireSat-Entwurfsprozesses. So sind zum Beispiel die Exploration des Entwurfsraums im Vorentwurf, der regelbasierte Aufbau des Geometriemodells und die Modellsynthese für die detaillierten Simulationen einige der Stationen im Entwurfsprozess, die mit hoher Detailtiefe umgesetzt wurden.

Es gibt jedoch auch Bereiche des Satellitenentwurfs, die nicht in diesem Beispiel enthalten sind. Dazu zählen zum einen im Entwurfsprozess vor- und nachgelagerte Tätigkeiten wie die Missionsanalyse [61] oder Assembly Integration and Test (AIT). Für diese beiden Beispiele wurde in den Arbeiten von Schaefer [61] für den Bereich der Orbitmodellierung und von Arnold [5] für die Fertigung bereits eine Machbarkeit prinzipiell nachgewiesen. Eine Modellierung dieser Tätigkeiten kann daher jederzeit ergänzend in die vorhandene Prozesskette eingefügt werden. Zum anderen sind einige Analysen des Detailentwurfs ausgespart, wie zum Beispiel eine Festigkeitsanalyse mittels eines FE-Modells oder die Überprüfung der elektromagnetischen Verträglichkeit (EMV). In dieser Arbeit wird mit den automatisch generierten Analysemodellen für die Thermalsimulation, Lageregelungssimulation und die Verkabelungssimulation die Machbarkeit der automatischen Modellsynthese an sich sichergestellt. Weitere Analysen können jederzeit ergänzend zu den bestehenden Entwurfssprachen hinzugefügt werden.

5.1 Ergebnisse

Mit der Beschreibung der Subsysteme für Datenverarbeitung, Lageregelung, Kommunikation, Struktur, Energieversorgung und der Nutzlast in den Vokabeln und Regeln der Entwurfssprachen besteht eine formale, computerlesbare Repräsentation des FireSat-Entwurfswissens. Die Integration dieser Entwurfssprachen erfolgt über die Satellitensystem-Entwurfssprache. Die Integration der geometrischen Aspekte erfolgt über die abstrakte Geometrierepräsentation in der Geometrie-Entwurfssprache. Mittels dieser Entwurfssprache wird die Beschreibung der Halterungen, der Versteifungstaschen und der Anordnung der Komponenten in weiteren

modular verwendbaren Entwurfssprachen ermöglicht. Aus der somit erzeugten abstrakten geometrischen Beschreibung werden ein detailliertes Geometriemodell, ein Thermalmodell und das Modell für das automatische Routing aus den Entwurfssprachen heraus generiert. Die weitere Beschreibung der Lageregelungssimulation erfolgt ebenfalls in einer Entwurfssprache und basiert auf den zuvor erzeugten Informationen. Durch die Repräsentation des gesamten Entwurfswissens in einer Hierarchie von Entwurfssprachen werden vier Probleme der Modellierung im Ingenieurwesen gelöst:

Die *Doppelarbeit* in verschiedenen Modellen fällt durch die Abbildung in einem zentralen Modell weg. So ist zum Beispiel die Einbauposition der PCDU des FireSat nur an einer Stelle modelliert. Verwendet wird diese Information im Geometriemodell, beim Erstellen der Taschen, im Thermalmodell und beim Routing. Auch die geometrische Definition der Box der PCDU des FireSat ist mit Hilfe der abstrakten Geometrieformulierung an einer zentralen Stelle vorgenommen und ersetzt die separate Definition derselben in CATIA V5, OpenCascade, ESATAN-TMS sowie die Definition bei der Netzerstellung für das Routing.

Das *Konsistenzproblem* wird durch die Generierung der Modelle aus einem konsistenten zentralen Modell heraus gelöst. Für die Konsistenz innerhalb des zentralen Modells werden die Parameter des Entwurfs über Gleichungen verknüpft. Die Konsistenz der Parameter innerhalb der Entwurfssprachen wird dann über die Lösung der mathematischen Gleichungen in einem Gesamtgleichungssystem sichergestellt. Die Konsistenz der Parameter der Entwurfssprachen mit den Parametern der Anwendungsprogramme wird durch die Generierung der Modelle für die Anwendungsprogramme sichergestellt. Das Geometriemodell in CATIA V5 und in OpenCascade wird aus der abstrakten Geometriedefinition mit einer einheitlichen Informationsbasis erstellt. Das Modell der Thermalsimulation in ESATAN-TMS beruht ebenfalls auf der Definition der abstrakten Geometrie. Des weiteren werden in diesem Modell die Informationen über den Energieverbrauch der verschiedenen Bauteile verwendet. Dies sind dieselben Werte, die für die Erstellung des Energiebudgets verwendet werden. Die Orbitdefinition der Simulation wird ebenfalls aus dem zentralen Modell ausgelesen. Damit sind die Werte der Thermalsimulation konsistent zu den Werten des zentralen Modells. Die Lageregelungssimulation in Simulink wird ebenfalls aus dem zentralen Modell generiert und ist damit konsistent zu diesem. Die für die Simulation verwendeten Komponenten des Lageregelungssystems werden aus der Entwurfssprache für das Lageregelungssystem ausgelesen. Andere Werte, wie zum Beispiel die Masse und das Trägheitsmoment des Satelliten, sind in der Satellitensystem-Entwurfssprache definiert und werden dort ausgelesen. Damit ist das Modell in Simulink ebenfalls konsistent zu dem zentralen Modell. Bei der Verwendung der Simulationsmodelle im Entwicklungprozess ist darauf zu achten, dass Änderungen in den Entwurfssprachen auch in die Anwendungsprogramme propagiert werden müssen und Änderungen in den Simulationsmodellen in die Entwurfssprachen aufgenommen werden müssen, um die Konsistenz der Modelle zu wahren.

Die *Automatisierung* von Routinetätigkeiten wird durch die Datenhaltung in einem zentralen Modell in größerem Umfang als bisher unterstützt. Dies wurde in dieser Arbeit durch die Automatisierung des gesamten Entwurfsprozesses des FireSat-Satelliten demonstriert. Der abstrakte Aufbau des FireSat-Entwurfs ist mittels Regeln und Programmen automatisiert. Die Umsetzung

dieser Automatisierung wurde in Abschnitt 4.1 und 4.2 ausführlich beschrieben. Die Automatisierung von konstruktiven Routinetätigkeiten wird anhand dem Anbringen von Halterungen in Abschnitt 4.4.2 und anhand der Konstruktion von Versteifungstaschen in Abschnitt 4.4.4 gezeigt. Durch die Einbindung des automatischen Routings wird eine neuartige Möglichkeit der Automatisierung demonstriert. Dieser Automatisierungsschritt ist nur auf der Basis der zentralen Datenhaltung mittels Entwurfssprachen umsetzbar. Diese und weitere Beispiele in dieser Arbeit zeigen Automatisierungsschritte in verschiedenen Bereichen des Satellitenentwurfs und die erweiterten Möglichkeiten zur Automatisierung mittels Entwurfssprachen sind dadurch demonstriert.

Die *Wiederverwendbarkeit* ist durch die Verwendung von Vokabeln und Regeln ganzheitlich für Komponenten, Subsysteme und auch Konstruktionsvorgänge möglich. Dies wird durch die in dieser Arbeit vorgestellten, flexibel einsetzbaren Entwurfssprachen gezeigt. Die Verwendung der verschiedenen Entwurfssprachen basiert auf der Hierarchie von der missionspezifischen FireSat-Entwurfssprache ganz unten bis hin zur abstrakten Geometrie-Entwurfssprache als oberstes Element der Hierarchie. Als Beispiel für die Wiederverwendbarkeit von Komponenten und Subsystemen kann das Kommunikationssystem mit seinen Antennen genannt werden. Diese Entwurfsspache kann für jede beliebige Satellitenmission wiederverwendet werden. Als Beispiel für die Wiederverwendung von Konstruktionsvorgängen können die automatisch erzeugten Versteifungstaschen genannt werden. Diese sind auf Basis der Halterungs- und Geometrieentwurfssprache jederzeit wiederverwendbar.

Die in dieser Arbeit vorgestellten Entwurfssprachen bilden die Grundlage für die Untersuchung der unterschiedlichen FireSat-Entwürfe. Die Untersuchungen des Kommunikationssystems in Abschnitt 4.1 und die Untersuchungen des Gesamtsystems in Abschnitt 4.3 zeigen eine Exploration von Entwurfsräumen auf Komponenten-, Subsystem- und Systemebene. Die regelbasierte Exploration des Entwurfsraums wie in Abb. 4.54 zeigt den Vergleich von über 3300 verschiedenen, vollständig analytisch ausgelegten Satelliten. Dafür wurde das Gleichungssystem des Entwurfs in jedem Punkt des Entwurfsraums für jede Variante des Kommunikationssystem erstellt und nach der Masse aufgelöst. Durch Rückeinsetzen der Gleichungen kann die Funktion der Masse in Abhängigkeit der Randbedingungen, in dem gezeigten Fall der Auflösung und Wellenlänge des Teleskops, berechnet werden. Die automatische Sensitivitätsanalyse mit den normierten Ableitungen der Variablen nach den Randbedingungen bietet zusätzlich die Möglichkeit alle Entwurfsparameter in einem Auslegungspunkt zu analysieren. Die aus den Ableitungswerten generierte Heatmap gibt Aufschluss über die Kopplungen zwischen den verschiedenen Systembereichen sowie über die Stärke der Kopplungen zwischen den Parametern.

Auf Basis der abstrakten Repräsentation aller Daten in einem Modell werden ein Geometriemodell, eine Regelungssimulation, eine Thermalsimulation und das Routingmodell für die automatische Verkabelung generiert. In Abschnitt 4.5 werden für drei parametrisch und topologisch unterschiedliche Varianten des FireSat-Beispiels die vollständigen Modelle für diese Simulationen erzeugt. Das Geometriemodell, die Modelle für die Thermalsimulation, die Simulink-Simulation und die Verkabelungsinformationen stellen für sich genommen in einer

herkömmlichen Entwicklungsumgebung einen Arbeitsaufwand von mehreren Tagen oder Wochen[1] dar. Die Verkabelung im CAD wird heutzutage in kleineren Projekten wegen des hohen Arbeitsaufwandes nicht durchgeführt. All diese Modelle können aus der Entwurfssprache „auf Knopfdruck“ und innerhalb von Minuten[2] erzeugt werden.

Durch die Verwendung der Entwurfssprachen ist der gesamte Entwurfsprozess in eine einheitliche Methodik eingebettet. Alle Entwurfsschritte in diesem Beispiel sind durch die Entwurfssprachen automatisch ausführbar. In Abschnitt 4.5 wird durch die vollständige Auslegung der Variante 1 und 2 des FireSat gezeigt, wie zwei unterschiedliche Systemtopologien im selben Auslegungspunkt untersucht werden können. Mit Variante 3 wird gezeigt, dass der gesamte Entwurfsprozess unter den geänderten Anforderungen neu ausgeführt werden kann. In Abschnitt 4.5.2 wird an der Auswahl der Drallräder gezeigt, dass auch in späten Entwurfsphasen Änderungen der Entwurfsregeln in vorhergehenden Entwurfsschritten ohne Probleme möglich sind. In Abschnitt 4.4.3 wird die Machbarkeit einer automatischen Anordnung dargestellt. Durch diese Beispiele wird demonstriert, dass mit dem Einsatz von Entwurfssprachen im Satellitenentwurf topologische und parametrische Änderungen entlang des gesamten Entwurfsprozesses auch in späten Entwurfsphasen ohne großen Aufwand möglich sind.

Der Anfangsaufwand für das Erstellen einer Entwurfssprache übersteigt den Aufwand für die manuelle Auslegung einer Systemtopologie in einem Entwurfspunkt bei weitem. Werden dann jedoch später topologische und parametrische Änderungen notwendig, Folgeprodukte entwickelt oder eine Baureihe abgeleitet, so zahlt sich der Mehraufwand sehr schnell aus. Durch die Wiederverwendbarkeit der Entwurfssprachen können auch einzelne Subsysteme oder Komponenten davon in neue Entwürfe integriert werden. Der Nutzen eines automatisierten Entwurfsprozesses wird anhand der Exploration des Entwurfsraums im Vorentwurf und anhand der generierten FireSat-Varianten in Abschnitt 4.5 unterstrichen.

Die Entwurfssprachen bieten für einen Satellitenentwurf in der Zukunft das Potential, die für einen Entwurf benötigte Zeit erheblich zu verkürzen. Die Zeit für den Entwurfsprozess beträgt für das FireSat-Beispiel auf einem handelsüblichen Computer[3] wenige Minuten[4]. Diese Zeit für den Entwurfsprozess steht, nach dem Aufbau der Entwurfssprachen, im Vergleich zu Wochen, Monaten oder sogar Jahren in einem herkömmlichen Entwicklungsprozess. Da die Entwurfszeit mit der Nutzung der gezeigten Entwurfssprachen nur noch die Summe der Programmlaufzeiten ausmacht, stellt dies hinsichtlich der gezeigten Modelle die theoretische Untergrenze für die Entwurfszeit dar.

[1] CATIA V5 Modell: ca. 1 Woche, Simulink Modell: ca. 2 Wochen, ESATAN-TMS Modell: ca. 2 Wochen, Verkabelung: ca. 2 Wochen. Bei diesen Angaben handelt es sich um optimistische Schätzungen des Autors, es steht jedoch außer Frage, dass es sich um andere Größenordnungen als bei der Generierung handelt.

[2] siehe Fussnote 4

[3] PC mit Intel Core2 Quad 2.5 Ghz, 8GB Ram.

[4] Ausführungszeiten für die FireSat-Variante 1 (die Zeiten für Variante 2 und 3 sind vergleichbar): Übersetzung der Entwurfssprache 3min 11sec, Erstellen des Geometriemodells in CATIA 6min 46sec /in OpenCascade 15sec, Generierung Thermalmodell für ESATAN 10sec, Generierung des Simulink-Modells 58sec, Verkabelung 2h30min.

5.2 Ausblick

Der gezeigte Entwurfsprozess bietet durch den modularen Aufbau eine grosse Bandbreite an Erweiterungsmöglichkeiten. Die Auslegung des FireSat in dieser Arbeit beginnt mit einer festgelegten Missionsbeschreibung ohne die Missionsanalyse zu berücksichtigen. Durch den Aufbau einer weiteren Entwurfssprache für die vorhergehende Beschreibung der Mission könnte der Entwurfsprozess um die Missionsanalyse erweitert werden. Dadurch könnte die Missionsanalyse in die Untersuchung des Entwurfsraums miteinbezogen werden und unter erweiterten Randbedingungen ein Optimum auf Missionsebene gesucht werden. Dies würde die Durchgängigkeit des Entwurfsprozesses erweitern.

Neben der Erweiterung in der Durchgängigkeit des Entwurfsprozesses können die vorhandenen Entwurfssprachen für eine breitere Untersuchung des Entwurfsraumes ausgebaut werden. Die Subsysteme des Satelliten können um weitere Lösungsprinzipien und Komponenten ergänzt werden. Es könnte zum Beispiel eine andere Satellitenstruktur verwendet werden oder in der Lageregelung ein System mit Triebwerken zum Einsatz kommen. Dadurch können die verschiedenen Entwurfssprachen so erweitert werden, dass neue Bereiche im Entwurfsraum abgedeckt werden und sie für eine grosse Bandbreite an Missionen wiederverwendbar sind.

Neben der Erweiterung der Entwurfssprachen sind auch die gezeigten Analysemöglichkeiten noch nicht ausgeschöpft. Für den in Abschnitt 4.3.4 dargestellten Entwurfsraum kann durch die Abbildung weiterer Dimensionen des Entwurfsraums ein umfassenderes Bild zu den verschiedenen Aspekten der FireSat-Auslegung geschaffen werden. Existiert eine analytisch formulierte Kopplung zwischen zwei Variablen des Entwurfs, so kann diese graphisch dargestellt werden. Damit können neue Erkenntnisse über den Entwurfsraum des FireSat gewonnen werden.

Die Heatmaps aus Abschnitt 4.3.5 können für weitere Untersuchungen des Systems herangezogen werden. Durch den Vergleich der Heatmaps an verschiedenen Auslegungspunkten eines Systems können unterschiedliche Sensitivitäten und damit „gutmütigere“ oder schwierigere Entwurfsbereiche festgestellt werden. Durch den Vergleich von Heatmaps verschiedener Systemtopologien kann der Einfluss der verwendeten Lösungsprinzipien auf die Kopplung der Variablen an die Randbedingungen untersucht werden. Damit könnte diese Technik für eine Entkopplung bestimmter Variablen von bestimmten Randbedingungen verwendet werden.

Für die Detailanalyse des Entwurfs können weitere Simulationsmodelle integriert werden und somit kann eine vollständigere Abdeckung des Satellitenentwurfs erreicht werden. Dadurch können weitere Kopplungen zwischen den verschiedenen Aspekten des Entwurfs aufgedeckt und untersucht werden. Mit der Integration einer Finite-Elemente Simulation könnten zum Beispiel die Auswirkungen der Materialwahl nicht nur auf die Thermal- sondern gleichzeitig auch auf die Festigkeitseigenschaften des FireSat automatisiert untersucht werden.

Der automatische Entwurf des FireSat Beispiel in dieser Arbeit lässt sich auch auf industrielle Entwürfe von Satelliten übertragen. Damit könnte auch für industrielle Entwürfe ein grösserer Bereich des Entwurfsraums nach guten Lösungen abgesucht werden. Dies ermöglicht in Zukunft den Entwurf besserer Satelliten in kürzerer Zeit.

Literaturverzeichnis

[1] AGUILAR, J., DAWDY, A., UND LAW, G. W. *The Aerospace Corporation´s Concept Design Center*. Proc. of the 8th Annual International Symposium of the International Council on Systems Engineering, Vancouver, Kanada, 1998.

[2] ALBER, R., UND RUDOLPH, S. *On a grammar-based design language that supports automated design generation and creativity*. 5th IFIP WG5.2 Workshop on Knowledge Intensive CAD, Malta, 23.-25. Juli, 2002.

[3] ALBER, R., RUDOLPH, S., UND KRÖPLIN, B. *On Formal Languages in Design Generation and Evolution*. Fifth World Congress on Computational Mechanics, Wien, Österreich, 12.-17. Juli, 2002.

[4] ANDERL, R., UND TRIPPENER, D. *STEP, STandard for the Exchange of Product Model Data.* Teubner Verlag, 2000.

[5] ARNOLD, P., UND RUDOLPH, S. *Bridging the gap between product design and product manufacturing by means of graph-based design languages*. 9th Interantional Symposium on Tools and Methods of Competitve Engineering (TMCE 2012), Karlsruhe, 7.-11. Mai, 2012.

[6] BALANIS, C. *Antenna Theory Analysis and Design*. John Wiley and Sons, Hoboken, New Jersey, USA, 2005.

[7] BANDECCHI, M., MELTON, B., GARDINI, B., UND ONAGRO, F. *The ESA/ESTEC Concurrent Design Facility*. Proc. of the 2nd European Systems Engineering Conference, München, S. 329-336, 2000.

[8] BEILSTEIN, L. *Eine Methodik zur analytischen Gewichtsabschätzung und Bewertung von Strukturverbindungen im Flugzeugvorentwurf.* Dissertation, Insitut für Flugzeugbau, Universität Stuttgart, 2012.

[9] BÖLLING, M. *Lösungspfad-basierte Analysen im Entwurf komplexer Systeme*. Dissertation, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2013 (eingereicht April 2013).

[10] DASSAULTSYSTEMS. *CATIA V5*. www.dassault-systems.com, 2013.

[11] DE KONING, H., EISENMANN, H., UND BANDECCHI, M. *Evolving Standardization Supporting Model Based Systems Engineering*. SECESA Conference Lausanne, Schweiz, 17.-19. Oktober, 2010.

[12] DELP, C. L. *The challenge of model-based systems engineering for space systems*. INCOSE INSIGHT Vol. 11 Issue 5: 14-18., 2008.

[13] DELP, C. L. *The challenge of model-based systems engineering for space systems, Year 2*. INCOSE INSIGHT Vol. 12 Issue 4: 36-39., 2009.

[14] ECSS. *Space Project Management, Project Phasing and Planning*. ESA Publications Division, ESTEC, P.O. Box 299, 2200 AG Noordwijk, Niederlande, 1996.

[15] ECSS. *Space Engineering - Engineering Design Model Data Exchange (CDF)*. E-TM-10-25A, ESA Requirements & Standards Division, ESTEC, P.O. Box 299, 2200 AG Noordwijk, Niederlande, 2010.

[16] ECSS. *Space Engineering - Space System Data Repository*. E-TM-10-23A, ESA Requirements & Standards Division, ESTEC, P.O. Box 299, 2200 AG Noordwijk, Niederlande, 2010.

[17] EHEIM, M. *Automatisches Routing und Packaging mit Entwurfssprachen*. Dissertation, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2013 (in Vorbereitung).

[18] EISENMANN, H., MIRO, J., UND DE KONING, H. P. *MBSE for European Space-Systems Development*. INCOSE INSIGHT Vol. 12 Issue 4: 47-53., 2009.

[19] GROSS, J., MESSE, C., UND RUDOLPH, S. *A Model Based Thermal Systems Engineering Approach*. SECESA Conference Lissabon, Portugal, 17.-19. Oktober, 2012.

[20] GROSS, J., UND RUDOLPH, S. *Hierarchie von Entwurfsentscheidungen im modellbasierten Entwurf komplexer Systeme*. Systems Engineering Konferenz "Tag des Systems Engineering"(TdSE 2011), Hamburg, 9.-11. November, 2011.

[21] GROSS, J., UND RUDOLPH, S. *Dependency Analysis in Complex System Design using the FireSat example*. INCOSE Symposium, Rom, Italien, 7.-12. Juli, 2012.

[22] GROSS, J., UND RUDOLPH, S. *Generating Simulation Models from UML - A FireSat Example*. 2012 Spring Simulation Multiconference, Orlando, Florida, USA, 26.-29. März, 2012.

[23] GROSS, J., UND RUDOLPH, S. *Regelbasierte Analyse von Entwurfsentscheidungen im Entwurf komplexer Systeme*. Systems Engineering Konferenz "Tag des Systems Engineering"(TdSE 2012), Paderborn, 7.-9. November, 2012.

[24] GÖTTLER, H. *Graphgrammatiken in der Softwaretechnik*. Springer Verlag, 1987.

[25] HAQ, M., UND RUDOLPH, S. *EWS-Car: A Design Language for Conceptual Car Design.* VDI-Berichte 1846, S. 213-237, Conference on Numerical Analysis and Simulation in Vehicle Engineering, Würzburg, Deutschland, 2004.

[26] IILS. *DesignCompiler 43*. www.iils.de, zuletzt besucht 01.02.2013, IILS mbH, 2005.

[27] IRANI, M., UND RUDOLPH, S. *Design Grammars for Conceptual Designs of Space Stations*. Proceedings of IAC International Astronautical Congress, IAC-03-T.P.02, Bremen, Deutschland, 2003.

[28] IRANI, M., UND RUDOLPH, S. *Space Station Design Rules*. SAE Aerospace Engineering Magazine, 10, S. 43-45, 2005.

[29] ITPENGINES. *ESATAN-TMS*. http://www.esatan-tms.com/, zuletzt besucht 01.02.2013.

[30] KARBAN, R., HAUBER, R., UND WEILKENS, T. *MBSE in Telescope Modeling*. INCOSE INSIGHT Vol. 12 Issue 4: 24-31., 2009.

[31] KARK, K. W. *Antennen und Strahlungsfelder*. Vieweg und Teubner, Wiesbaden, 2010.

[32] KOCAK, M. S. *Erstellung einer Schnittstelle zur generischen Thermalsimulation von Satelliten*. Diplomarbeit, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2010.

[33] KORMEIER, T. *Graphenbasierte Entwurfssprachen zur konsistenten Modellierung und musterbasierten Topologiemodifikation von Faserverbundstrukturen*. Dissertation, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2010.

[34] KORMEIER, T., ALBER, R., UND RUDOLPH, S. *Topological and parametrical design of aircraft surfaces using design grammars*. Proc. DGLR Symposium Deutscher Luft- und Raumfahrtkongress, München, Deutschland, 2003.

[35] KORNFELD, G. K. *Satellite TWT-Amplifiers using the Microwave Power Module Concept and Radiation Cooled Collectors*. IEDM Seite 787-790, 1994.

[36] KRÖPLIN, B., UND RUDOLPH, S. *Entwurfsgrammatiken - Ein Paradigmenwechsel?* Der Prüfingenieur, 2005.

[37] KUHNE. *www.kuhne-electronic.de*. zuletzt besucht 15.03.2012, Kuhne electronic GmbH, 2012.

[38] LANDES, B. *On physical aspects of cabin architectures using tolerancing methods*. Dissertation, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2013.

[39] LANDES, B., UND RUDOLPH, S. *Aircraft cabin architectures including tolerancing using a graph-based design language in UML.* Proceedings Deutscher Luft- und Raumfahrt Kongress 2011, Bremen, 27.-29. September, 2011.

[40] LARSON, W. J., Ed. *Applied Space Systems Engineering*. Space Technology Series. McGraw Hill, 2009.

[41] LINDENMAYER, A. *Mathematical Models for cellular interaction in development*. Part 1 and 2. Journal of Theoretical Biology, 18, S. 280-315, 1968.

[42] MATHWORKS. *Matlab*. http://www.mathworks.de/products/matlab/, zuletzt besucht 01.02.2013.

[43] MATHWORKS. *Matlab Simulink Aerospace Toolbox*. http://www.mathworks.de/products/-aerotb/, zuletzt besucht 01.02.2013.

[44] MILDNER, C. *Regelbasierte Erstellung von Sicken zur Gewichtsreduktion von Montageplatten*. Hochschule Reutlingen, 2012.

[45] NOSER, H.-R., RUDOLPH, S., UND STUCKI, P. *Physics-Enhanced L-Systems*. Proc. 9th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2001), Pilsen, Tschechische Republik, 5.-9. Februar, 2001.

[46] OMG. *OMG Systems Modeling Language (OMG SysML)*. OMG, Version 1.2, OMG Document Number: formal/2010-06-01, http://www.omg.org/spec/SysML/1.2/ , Juni, 2010.

[47] OMG. *OMG Unified Modeling Language (OMG UML), Superstructure, V2.4.1*. OMG Document Number: formal/2009-02-02, http://www.omg.org/spec/UML/2.2/Superstructure, August, 2011.

[48] OMG. *OMG Object Constraint Language (OCL)*. Version 2.3.1, OMG Document Number: formal/2012-01-01, http://www.omg.org/spec/OCL/2.3.1, 2012.

[49] OPENCASCADE. *www.opencascade.org*. zuletzt besucht 01.02.2013, 2013.

[50] PAHL, G., UND BEITZ, W. *Konstruktionslehre*. 6. Auflage, Springer Verlag, Berlin, 2005.

[51] PEAK, R., PAREDIS, C., MCGINNIS, L., UND FRIEDENTHAL, S. *Integrated System Design with Simulation and Analysis Using SysML.* INCOSE INSIGHT Vol. 12 Issue 4: 40-44., 2009.

[52] REICHWEIN, A. *Application-specific UML profiles for multidisciplinary product data integration*. Dissertation, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2011.

[53] RIESTENPATT GENANNT RICHTER, M. *Eine Entwurfssprache zur Auslegung der Lage- und Bahnregelung von Satelliten*. Studienarbeit, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2011.

[54] ROMBERG, O., BRAUKHANE, A., UND SCHUMANN, H. *Status of the Concurrent Engineering Facility at DLR Bremen*. Deutscher Luft- und Raumfahrt Kongress (DLRK), Darmstadt, 2008.

[55] RUDOLPH, S. *Übertragung von Ähnlichkeitsbegriffen*. Habilitationsschrift, Universität Stuttgart, 2002.

[56] RUDOLPH, S. *Know-How Reuse in the Conceptual Design Phase of Complex Engineering Products - Or: Are you still constructing manually or do you already generate automatically?* Conf. Proc. Integrated Design and Manufacture in Mechanical Engineering, Grenoble, Frankreich, 17.-19. Mai, 2006.

[57] RUDOLPH, S. *Skript zur Vorlesung Digitaler Produktentwurf*. am Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2010.

[58] RUDOLPH, S., UND ALBER, R. *An Evolutionary Approach to the Inverse Problem in Rule-Based Design Representations*. Proc. of the 7th. International Conference on Artificial Intelligence in Design (AID 2002), Cambridge University, Cambridge, UK, 15.-17. Juli, 2002.

[59] RUDOLPH, S., UND BOELLING, M. *Constraint-based conceptual design and automated sensitivity analysis for airship concept studies*. Aerospace Science and Technology 8,S. 333-345, 2004.

[60] RUDOLPH, S., UND NOSER, H. *On Engineering Design Generation with XML-based Knowledge-Enhanced Grammars*. Proceedings IFIP WG5.2Workshop on Knowledge Intensive CAD Parma, Italien, 22.-24. Mai, 2000.

[61] SCHAEFER, J. *Eine Entwurfsgrammatik zur Flugbahnentwicklung und Bewertung für Raumfahrzeuge*. Diplomarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2003.

[62] SCHAEFER, J., ALBER, R., UND RUDOLPH, S. *Satellite Design by Design Grammars*. DGLR Jahrestagung, 2003.

[63] SCHAEFER, J., UND RUDOLPH, S. *Satellite design by design grammars*. Aerospace Science and Technology, 9, S. 81-91, 2005.

[64] SCHAEFER, J., UND RUDOLPH, S. *Spacecraft Thermal Model Generation by Design Compilation*. Deutscher Luft- und Raumfahrtkongress, Friedrichshafen, Deutschland, 2005.

[65] SCHMIDT, J. *Eine Entwurfssprache für Geometrie*. Studienarbeit, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2012.

[66] SMITH, J. *Concurrent Engineering in the Jet Propulsion Laboratory Project Design Center*. Society of Automotive Engineers, Inc., 98AMTC-83, 1997.

[67] USPRO. *Initial Graphics Exchange Specification IGES 5.3*. US Product Data Association (USPRO); http://www.uspro.org/documents/IGES5-3_forDownload.pdf, 1996.

[68] VOGEL, S., DANCKERT, B., UND RUDOLPH, S. *Entwicklung von Abgasnachbehandlungssystemen auf Basis einer graphenbasierten Entwurfssprache*. ATZ-MTZ Konferenz Heavy-Duty-, On- und Off-Highway-Motoren, Mannheim, 23.-24. November, 2010.

[69] VOGEL, S., DANCKERT, B., UND RUDOLPH, S. *Wissensbasierte Entwicklung von SCR-Systemen mit graphenbasierten Entwurfssprachen*. MTZ - Motortechnische Zeitschrift September 2012, Volume 73, Issue 9, pp 702-708, 2012.

[70] WERTZ, J., Ed. *Space Mission Analysis and Design*. Microcosm Press El Segundo, California, USA, 1999.

[71] WOLFRAMRESEARCH. *Mathematica*. http://www.wolfram.com/mathematica/, zuletzt besucht 01.02.2013.

[72] YUSAN, H., UND RUDOLPH, S. *On Systematic Knowledge Integration in the Conceptual Design Phase of Airships*. Proc. 13th AIAA Lighter-Than-Air Technology Conference, Norfolk, VA, 28.-30. Juni, 1999.

[73] YUSAN, H., UND RUDOLPH, S. *A Study of Constraint Management Integration in to the Conceptual Design Phase*. ASME Design Engineering Technical Conferences, Las Vegas, Nevada, USA, 12.-15. September, 1999.

[74] ZARM. *Magnetic Torquers for Spacecraft Control*. Zarm Technik AG, 2002.