Modell-basierte Systemsimulation eines Kleinsatelliten mit einem FPGA-basierten On-board Computer

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

vorgelegt von

Dipl.-Ing. Albert Falke

aus Lennestadt

Hauptberichter	Prof. Dr.-Ing. Hans-Peter Röser
Mitberichter	Hon.Prof. Dr.-Ing. Rudolf Benz
	Prof. Dr.-Ing. Klaus Brieß

Tag der mündlichen Prüfung: 02. März 2009

Institut für Raumfahrtsysteme der Universität Stuttgart

2009

Zusammenfassung

Das innovative Stuttgarter Kleinsatellitenprogramm des Instituts für Raumfahrtsysteme der Universität Stuttgart stellt sich der Aufgabe, Kleinsatelliten zur Technologie-Evaluation und Durchführung von wissenschaftlichen Experimenten zu bauen, zu starten und im Erdorbit zu betreiben. Dazu wird am Institut für Raumfahrtsysteme die Infrastruktur für den Bau und Betrieb des Satelliten geschaffen. Das Ziel des Programms ist es nicht nur die Kleinsatelliten im Erdorbit als abgeschlossenes Ergebnis zu betreiben. Vielmehr soll durch den gesamten Entwicklungsprozess der Satelliten wissenschaftlicher und technischer Nachwuchs ausgebildet und qualifiziert werden. Zusammenfassend stellt sich der Nutzen des Programms durch wissenschaftliche Beiträge der Experimente im Erdorbit, durch die Erprobung neuartiger Technologien in Satellitensystemen und die Schaffung eines Kompetenzzentrums für den Bau und Betrieb von Kleinsatelliten am Stuttgarter Institut für Raumfahrtsysteme dar.

Der Erdbeobachtungssatellit *Flying Laptop* wird als erstes entwickelt und ist mit der Technologiedemonstration als einem seiner Ziele ein wichtiges Element zur Vorbereitung der nachfolgenden Missionen. Wissenschaftliche Herausforderung ist die Vermessung der Bidirektionalen Reflektanzverteilungsfunktion mit optischen Kameras. Er besitzt einen quaderförmigen Körper mit einer Kantenlänge von durchschnittlich 70 cm bei einer angestrebten Gesamtmasse von weniger als 120 kg. Zielorbit des Kleinsatelliten ist ein polarer, sonnen-synchroner niedriger Erdorbit in 500-900 km Höhe. Der On-board Computer des *Flying Laptop* wird aus einem Field Programmable Gate Array (FPGA) basierten System bestehen. Gegenüber normalen Mikroprozessoren mit fester interner Logik kann die interne Logik des FPGA re-programmiert werden, um die optimierten bzw. erweiterten Funktionalitäten des On-board Computers abzubilden. Die Anwendung des FPGA-basierten Computers führt durch die hohe Parallelität der On-board Kontrollalgorithmen gegenüber einer konventionellen Prozessorarchitektur somit zu einer signifikant gesteigerten Leistung.

Die vorliegende Dissertation beschäftigt sich schwerpunktmäßig mit der Systemsimulation des Satelliten *Flying Laptop* unter Weltraumbedingungen, mit dem Ziel die Funktionalitäten des Satellitensystems als Ganzes zu verifizieren. Simulation bezeichnet allgemein das Nachahmen des Verhaltens eines Systems oder Prozesses zum Zwecke der Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu kompliziert sind. Auch im Kontext einer Kleinsatellitenmission ist das Satellitensystem bereits so komplex, dass man ohne die systemweite Simulation keine detaillierte Analyse unter Berücksichtigung der vielen miteinander interagierenden Komponenten mehr durchführen kann.

Bereits seit einigen Jahren verwenden renommierte Satellitenhersteller Simulationstechnologien, um bereits während der Satellitenentwicklung und Fertigung den größtmöglichen Missionserfolg zu gewährleisten. Eine dieser Technologien ist die Modell-basierte Entwicklungs- und Verifikationsumgebung der Firma EADS Astrium GmbH aus Friedrichshafen. Sie bietet ein systematisches und standardisiertes Entwicklungs- und Verifikationsrahmenwerk im Sinne eines systemweiten Satellitensimulators an, um die Entwicklung von Satelliten, die Verifikation der On-board Software und den Gesamtfunktionsnachweis des Satelliten zu unterstützen. In diesem Simulator können alle Satellitenkomponenten modelliert und die On-board Software somit auf funktionaler Ebene verifiziert werden. Auf diese Weise können aufwendige und kostenintensive Entwicklungsmodelle einzelner Subsysteme und Schlüsseltechnologien wegfallen. Insbesondere sei hier die Verifizierung der Genauigkeit des Lageregelungssystems angesprochen. Als technologische Voraussetzung bringt gerade diese

Entwicklungstechnologie budgetschwache Kleinsatellitenprojekte der Realisierung einen deutlichen Schritt näher und erhöht gleichzeitig die Güte des Systemdesigns des Satelliten. Im Rahmen einer Kooperation wurde diese Simulationsumgebung dem Institut für Raumfahrtsysteme zur Adaption und Anwendung im Stuttgarter Kleinsatellitenprogramm zur Verfügung gestellt.

Als Voraussetzung für die Durchführung von Systemsimulationen mit den von EADS Astrium bereitgestellten Simulatorkomponenten sind eine Reihe von Aufgaben zu bewältigen. Dazu gehören in erster Linie der Aufbau der Simulator-Infrastruktur mit der dazu erforderlichen Computer Hardware, der Erstellung eines projektbezogenen Datenbanksystems, die Schaffung eines Datenbankextraktionswerkzeugs zur Erzeugung stilkonformer XML-Konfigurationsdateien des Simulators, die Umsetzung eines Concurrent Versions System Repositoriums für die Quellcodeverwaltung und die Einrichtung eines UML-Entwicklungswerkzeugs. Weiterhin ist mit dem von der ESA angebotenem Missionskontrollsystem SCOS-2000 die Beschaffung, Installation, Konfiguration und Inbetriebnahme eines anerkannten Systems zur Steuerung und Überwachung von Systemsimulationen realisiert worden.

Zur Anwendung der Systemsimulation im Kleinsatellitenprojekt *Flying Laptop* sind alle relevanten Satellitenkomponenten im Simulator durch entsprechende, detaillierte Softwaremodelle abgebildet worden. Der sukzessiven Entwicklung der Komponentenmodelle ist durch den schrittweisen Aufbau von immer komplexeren Teststänten der Software-Verifikations-Einrichtung Rechnung getragen worden. Infolgedessen erweiterten sich die Simulationsfähigkeiten von ersten Orbitsimulationen mit wenigen Komponentenmodellen, aber mit geschlossenem Simulationskreislauf, über die Simulation von Energie- und Thermalbilanzen unter Betriebsbedingungen bis hin zu vollständigen Systemsimulationen unter Einbindung der realen On-board Kontrollalgorithmen auf einem FPGA-Entwicklungsboard. Gerade dieses charakteristische FPGA-basierte On-board Computer System mit den darauf betriebenen Kontrollalgorithmen führt zu einer großen Herausforderung bei der modellhaften Repräsentation derselbigen im Systemsimulator. In diesem Zusammenhang stellt die Einbindung der realen On-board Kontrollalgorithmen auf einem FPGA-Entwicklungsboard in den geschlossenen Simulationskreislauf eine erfolgreich realisierte technische Neuerung dar.

Zum Zeitpunkt der Fertigstellung dieser Arbeit steht dem Projekt *Flying Laptop* durch die Einbindung der realen On-board Kontrollalgorithmen auf dem FPGA-Entwicklungsboard ein umfangreicher Teststand mit großem Potential zur Entwicklung und zum Funktionsnachweis der On-board Kontrollalgorithmen zur Verfügung. Dieser wurde, wie die vorgestellten Simulationsergebnisse zeigen, bereits intensiv zum Testen der ersten Lageregelungsalgorithmen genutzt und kann auch zukünftig sukzessive mit Funktionserweiterungen der On-board Kontrollalgorithmen angewandt werden. Typische Missionsszenarien wie die Transition zwischen zwei Betriebsmodi des Satelliten oder der automatische Übergang in den `SAFE` Mode bei einem Fehler können jetzt simuliert und getestet werden. Die Ergebnisse dieser Simulationen dienen nicht nur dem reinen Funktionsnachweis der On-board Kontrollalgorithmen, sondern fließen direkt in die Optimierung der Kontrollalgorithmen zurück und führen so zu einem iterativem Verbesserungsprozess. Der Systemsimulator ist mit seiner Datenbank als Teil seiner Infrastruktur ganz gezielt so implementiert worden, dass einzelne Testszenarien einfach und ohne wiederholten Konfigurationsaufwand reproduziert werden können. Somit unterstützt der Systemsimulator den sich in der Softwareentwicklung typischerweise iterativ wiederholenden Verifikationsprozess in einer optimalen Form.

Abstract

Model-based system-simulation of a micro-satellite with a FPGA-based On-board Computer

The innovative “Stuttgart Small Satellite Programme” of the Universität Stuttgart (Institute of Space Systems) has the objective to design, build, launch and operate small satellites for technology evaluation and accomplishment of scientific experiments. The infrastructure to build and operate the satellite in its orbit is therefore setup at the Institute of Space Systems. Common goal of the programme is not only as results the small satellites in their orbits, but also the education and qualification of junior scientists and engineers throughout the development process of the satellites. In summary the programme benefits are represented by scientific contributions of the experiments in orbit, by successful verification of new technologies applied to the satellite systems and by the established center of competence for building and operating small satellites at the Institute of Space Systems in Stuttgart.

The earth observation satellite *Flying Laptop* will be developed first. With its technology demonstration of new system components as one of its mission objectives, the *Flying Laptop* is an important milestone for the subsequent missions. Scientific challenge of this mission is the measurement of the bi-directional reflectance distribution function with an optical camera system. While the overall satellite mass shall be less than 120 kg the *Flying Laptop* is designed with a cubical shape of an average of 70 cm edge length. The aspired orbit is a polar, sun-synchronous low earth orbit of 500-900 km height. The on-board computer of the *Flying Laptop* will consist of a system, which is based on Field Programmable Gate Arrays (FPGA). The high computational power and the strong parallelism of the on-board control algorithms lead to significantly increased performance of the FPGA-based computer system compared to conventional processor architectures.

This PhD thesis treats the topic of system-simulation of the satellite *Flying Laptop*. Its intention is verification of the satellite functionalities as a complex system in space environment. The idea of simulation characterizes in a common meaning the emulation of the behavior of a system or process in order to analyze the complex system, which cannot represented by analytic mathematics. In context of a small satellite mission the satellite system is already a complex system. Without system-wide simulation a detailed analysis of the satellite system with consideration of all interacting components cannot be accomplished.

Since several years renowned satellite manufacturers utilize simulation technologies to guarantee maximized mission success already during the satellite development, integration and test. One of this technologies is the Model-based Development and Verification Environment (MDVE), which is developed by EADS Astrium GmbH in Friedrichshafen, Germany. By meanings of a system-wide satellite simulator it provides a systematic and standardized development and verification framework in order to support the development process of satellites, the verification of the on-board software and to support the overall system verification with all individual component functionalities. In the simulated space environment all satellite components are represented by software models and the on-board software is thereby verified on functional level. In this matter complex and expensive engineering models of individual subsystems and key technologies can be dropped. The verification of the attitude control system performance is to be mentioned in this context in particular. As a technological premise this development technology on the one hand helps small satellite projects with

limited budgets to realize their mission objectives and on the other hand increases the satellite system design and verification quality. The simulation environment MDVE was provided to the Institute of Space Systems for its small satellite projects in the Stuttgart Small Satellite Programme due to a close cooperation with EADS Astrium.

To setup the simulation environment with the provided simulator parts by EADS Astrium and to accomplish first system-simulations, a couple of tasks needed to be done. First and foremost the simulator infrastructure was setup on the corresponding computer hardware: the project specific database system must be configured and set up, a software application for database extraction and generation of style-conform XML simulator configuration files must be created, a Software Version Control repository must be implemented and finally the UML development software was installed and configured. Furthermore the acquisition, installation, configuration and commissioning of the mission control system SCOS-2000, provided by ESA, establishes an accredited system for simulation control and monitoring.

For system-simulation application in the small satellite project *Flying Laptop* all relevant satellite components are represented by corresponding detailed software models. The successive development of these component models ends up in the stepwise setup of even more complex Software Verification Facilities. Hence the simulation abilities expand from first closed loop orbit simulations with only a few component models, over simulations of power and thermal balance at operating conditions to complete system simulations with implementation of the real on-board control algorithms on a FPGA development board. Specifically the characteristic FPGA-based On-board Computer System with its control algorithms lead to a big challenge by representing them in the system-simulator. In this context a technical innovation is introduced by the successful implementation of the real on-board control algorithms on a FPGA development board into the closed simulation loop.

By completion of this PhD thesis with implementation of the real on-board control algorithms into the simulation loop, the project *Flying Laptop* is supported with a comprehensive and highly potential testbed for development and functional verification of the on-board control algorithms. This testbed is eager used to test the already developed attitude control algorithms as first simulation results demonstrate. It also can be used for further developments and optimizations of the on-board control algorithms. Typical mission scenarios which may be simulated and tested are the transition between two operational modes of the satellite or the autonomous switch into `SAFE` mode due to an arbitrary injected malfunction. The results of these simulations serve not only as functional verification of the on-board control algorithms but can also analyzed and traced back to the optimization of these control algorithms and lead to an iterative process of improvement. With its database system as part of its infrastructure, the system simulator is implemented systematically to easily support the reproduction of individual test cases without the need of a repeated complex and time-consuming configuration. In this manner the on-board software development with its repeated verification processes is supported in an ideal way.

(Further informations can be found in the authors publications and referred journal articles enclosed in the annex of this thesis.)

Inhaltsverzeichnis

0 Einleitung 1

1 Einordnung der Aufgabenstellung 2

2 Grundlagen und Basistechnologien 4

2.1 Der Kleinsatellit Flying Laptop im Rahmen des Stuttgarter Kleinsatelliten Programms 4

2.1.1 Der erste Stuttgarter Kleinsatellit: Flying Laptop 5

2.1.2 Systemkomponenten des Flying Laptop 7

2.2 Theorie der Simulation 14

2.2.1 Der Begriff Simulation 14

2.2.2 Typen von Simulationen und Zeitfortschrittsmechanismen 15

2.3 Modell-basierte Entwicklungs- und Verifikationsumgebung 17

2.3.1 Modell-basierte Systemsimulationsumgebung in der Satellitenentwicklung – Ein Überblick der wichtigsten Elemente und Funktionalitäten 19

2.3.2 Teststand Konfigurationen der Systemsimulationsumgebung 25

2.4 Softwaretechnische Voraussetzungen 28

2.4.1 Concurrent Versions System 28

2.4.2 Datenbanksysteme 28

2.4.3 Extensible Markup Language 29

2.4.4 Unified Modeling Language 29

2.4.5 Wiki 30

3 Durchführung und Umsetzung 33

3.1 Hard- und Software Entwicklungs-Infrastruktur 33

3.1.1 Computer Hardware 34

3.1.2 Grundlegende Software Installation 35

3.1.3 Das Flying Laptop Concurrent Versions System Repositorium 36

3.1.4 Die Flying Laptop Projekt Datenbank 37

3.1.5 Automatisierte Erzeugung der Konfigurationsdateien des Simulators 37

3.2 Die Software Verification Facility und die Anwendung im Flying Laptop Projekt 39

3.2.1 Generelles zur Implementation des Systemsimulators am Institut für Raumfahrtsysteme der Universität Stuttgart 39

3.2.2 Definition und Entwicklungsstufen der Systemsimulator Teststände im Flying Laptop Projekt 41

3.3 Simulationsmodelle für den Flying Laptop System Simulator 45

3.3.1 On-board Computer Simulator mit FPGA Hardware 46

3.3.2 Modell des Sonnensensorsystems 55

3.4 Aufbau des zentralen Missionskontrollsystems 63

3.4.1 Selektion eines geeigneten Systems 63

3.4.2 Installation und Konfiguration 65

3.4.3 Verbindung des Missionskontrollsystems mit dem Systemsimulator 65

3.4.4 Bedienung von simuliertem Satellit und Systemsimulator 66

4 Darstellung der Ergebnisse 69

4.1 Aufbau einer hard- und softwaregestützten Entwicklungsumgebung 69

4.1.1 Implementation der Simulator-Infrastruktur 69

4.1.2 Erstellung von Komponentenmodellen des Echtzeit-Simulators 70

4.2 Simulationen von Missionsszenarien im Hinblick auf Lageregelung, Thermal- und Energiemanagement 72

4.2.1 Der DETUMBLE Mode des Lageregelungssystems 76

4.2.2 Der SAFE Mode des Lageregelungssystems 77

4.2.3 Identifikation des Referenzorbits mit den ungünstigsten Bedingungen für das Energieversorgungssystem 80

4.2.4 Vergleich der Solarpaneel-Konfigurationen im SAFE Mode des Satelliten 82

4.2.5 Vergleich der Solarpaneel Konfigurationen im IDLE Mode des Satelliten 84

4.2.6 Simulation von Separationsszenarien des Satelliten vom Orbittransferfahrzeug und seiner ersten Erdumkreisungen 85

4.3 Zusammenfassung der Ergebnisse 90

5 Ausblick 92

6 Literaturverzeichnis 94

Anhang i

Abbildungsverzeichnis

Abbildung 2.1: Modellzeichnung des Kleinsatelliten Flying Laptop.....5

Abbildung 2.2: Baugruppen und Module des Kleinsatelliten Flying Laptop.....6

Abbildung 2.3: Übersicht der Systemkomponenten, der elektrischen Schnittstellen und Kommunikationsverbindungen des Satelliten Flying Laptop.....8

Abbildung 2.4: Unterteilung der Projektphasen in Raumfahrtprojekten nach Benz [Benz '07] 18

Abbildung 2.5: Komponenten des MDVE Systemsimulators [Eickhoff '08].....19

Abbildung 2.6: Interne Struktur des Echtzeit-Simulator-Moduls [Eickhoff '08].....20

Abbildung 2.7: Ablaufreihenfolge und Datenfluss zwischen Modellen des Lageregelungssystems; hier am Beispiel der Systemkomponenten des Kleinsatellitenprojekts Flying Laptop 24

Abbildung 2.8: MDVE Konfiguration: Software-Verifikations-Einrichtung [Eickhoff '08]…26

Abbildung 2.9: MDVE Konfiguration: Echtzeit-Teststand [Eickhoff '08].....27

Abbildung 2.10: Die UML 2.0 Diagrammtypen in hierarchischer Darstellung [Ryan '07].....30

Abbildung 2.11: Startseite des MoonWiki unter http://moonwiki.irs.uni-stuttgart.de/.....31

Abbildung 3.1: Das MDVE-Labor am IRS mit farblich hervorgehobenem Teststand.....33

Abbildung 3.2: Graphische Darstellung der Computerhardware-Infrastruktur.....34

Abbildung 3.3: Ablauf der Erstellung und Integration eines Satelliten Komponentenmodells in den Systemsimulator mit den beteiligten Werkzeugen und Elementen graphisch dargestellt.....40

Abbildung 3.4: Graphische Darstellung der Komponenten der Software Verification Facility in der ersten Entwicklungsstufe SVF v0.1.....42

Abbildung 3.5: Graphische Darstellung der Software Verification Facility in der zweiten Entwicklungsstufe SVF v0.5.....43

Abbildung 3.6: Graphische Darstellung der Software Verification Facility in der letzten und vollständigen Entwicklungsstufe SVF v1.0.....44

Abbildung 3.7: Schematische Struktur der Entwicklungs Software Verification Facility mit Echtzeit-Simulator (RTS) und implementiertem On-board Computer Simulator (OBC-SIM).....47

Abbildung 3.8: Strukturdefinition des Simulator-Transfer-Frames.....48

Abbildung 3.9: UML Sequenzdiagramm der Sequentiellen Ablaufsteuerung.....52

Abbildung 3.10: UML Sequenzdiagramm des parallelen Synchronisationsprinzips.....53

Abbildung 3.11: Position und Orientierung der Solarzellensensoren an der Satellitenstruktur 55

Abbildung 3.12: Nicht maßstäbliche Skizze der geometrischen Verhältnisse der Albedo Reflektion.....58

Abbildung 3.13: Struktur und Kommunikationsverbindungen des Sonnensensorsystems.....62

Abbildung 3.14: SCOS-2000 mit ein paar Beispielen von Telemetrie Visualisierungsfenstern 64

Abbildung 3.15: Verbindungselemente zwischen dem Missionskontrollsystem SCOS-2000 und dem MDVE Systemsimulator.....66

Abbildung 3.16: SCOS-2000 mit Fenstern zur Auswahl und Absendung von Telekommandos 67

Abbildung 3.17: Datenflüsse und Verbindungsarten zwischen den beteiligten Elementen von Missionskontrollsystem, Echtzeit-Simulator und On-board Computer Simulator.....68

Abbildung 4.1: Das FPGA-Entwicklungsboard „RC10“ der Firma Celoxica mit modifizierter serieller Schnittstelle.....71

Abbildung 4.2: Bedingungen des Bewegungszustandes und der relativen Lage des Satelliten Flying Laptop zur Sonne in den Betriebsmodi SAFE und IDLE.....73

Abbildung 4.3: Maßstäbliche Illustration der geometrischen Lage der Referenzorbits.....74

Abbildung 4.4: Die eingeführten Solarpaneel-Konfigurationen A, B und C.....75

Abbildung 4.5: Induziertes Drehmoment und resultierende Drehraten im DETUMBLE Mode 76

Abbildung 4.6: Induziertes Drehmoment und resultierende Drehraten im SAFE Mode.....78

Abbildung 4.7: Orientierungswinkel βSun der Solarpaneele zur Sonne im SAFE Mode.....79

Abbildung 4.8: Vorläufige Optimierungen des SAFE Mode Lageregelungsalgorithmus.....79

Abbildung 4.9: Einflüsse der Orbitparameter auf die Energiebilanz.....81

Abbildung 4.10: Energiebilanz im SAFE Mode in den Solarpaneel-Konfigurationen A, B & C unter ungünstigsten Simulationsbedingungen für das Energieversorgungssystem.....83

Abbildung 4.11: Energiebilanz im IDLE Mode in den Solarpaneel-Konfigurationen A, B & C unter ungünstigsten Simulationsbedingungen für das Energieversorgungssystem.....85

Abbildung 4.12: Induziertes Drehmoment und resultierende Drehraten für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug.....87

Abbildung 4.13: Orientierungswinkel βSun der Solarpaneele zur Sonne für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug.....88

Abbildung 4.14: Energiebilanz für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug.....89

Abbildung 4.15: Graphische Zusammenstellung der Komponenten und Aufgaben zur Durchführung von Systemsimulationen; Die Leistungen des Autors sind in blau hervorgehoben 90

Tabellenverzeichnis

Tabelle 3.1: Benötigte und installierte Softwarepakete auf den MDVE-Labor Computern.…36

Tabelle 3.2: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v0.1 42

Tabelle 3.3: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v0.5 43

Tabelle 3.4: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v1.0 44

Tabelle 3.5: Liste der erstellten Simulationsmodelle für den Kleinsatelliten Flying Laptop...45

Tabelle 3.6: Definition der Feldlängen des Simulator-Transfer-Frames.....49

Tabelle 3.7: Liste der definierten Kontrollframes.....50

Tabelle 3.8: Steuerzeichen Ersetzungstabelle des DLE Protokolls.....51

Tabelle 3.9: Telemetrie Format des Sonnensensorsystems.....62

Tabelle 4.1: Zusammenfassung und Integrationszeitpunkt aller erstellen Softwaremodelle…70

Tabelle 4.2: Verweildauer des Satelliten in der Sonne, im Erdschatten und im Halbschatten in den jeweiligen Referenzorbits.....75

Abkürzungen und Akronyme

&	und
bzw.	beziehungsweise
ca.	circa
d. h.	das heißt
et al.	et alii (lateinisch für „und andere“)
evtl.	eventuell
ggf.	gegebenenfalls
u. a.	und andere, unter anderem
usw.	und so weiter
z. B.	zum Beispiel

ACS	Attitude Control System (Lageregelungssystem)
ADC	Analogue/ Digital Converter (Analog/ Digitalwandler)
API	Application Programming Interface
APID	Application Process Identification
BOL	Begin-of-Life (Beginn der Lebenszeit)
BRDF	Bi-directional reflectance distribution function (Bi-direktionale Reflektanzverteilungsfunktion)
C/A	Control Algorithm (Kontrollalgorithmus)
C&DH	Command and Data Handling System (Kommando- und Datenverarbeitungssystem)
CCS	Central Control System (Zentrales Kontrollsystem)
CCSDS	Consultative Committee for Space Data Systems
CDR	Critical Design Review
CDT	C/C++ Development Tools (C/C++ Entwicklungswerkzeuge)
CDV	Command Decoder and Voter (Kommando Dekodier- und Entscheidungseinheit)
CHK	Checksumme (Prüfsumme)
CPN	Central Processing Node (Zentraler Prozessierungsknoten)
CR	Carriage Return (Zeilenumschaltung)
CVS	Concurrent Versions System (Parallelentwicklungs- und Versionsverwaltungssystem)
DBMS	Database Management System (Datenbankmanagementsystem)
DLE	Data Link Escape (Datenübertragungsumschaltung)
DLR	Deutsches Zentrum für Luft- und Raumfahrt e. V.
DPU	Data Processing Unit (Datenprozessierungseinheit)
DTD	Document Type Definition

EADS	European Aeronautic Defense and Space Company
EGSE	Electrical Ground Support Equipment
EOCV	End-of-Charge-Voltage (Ladeschlussspannung)
EOL	End-of-Life (Ende der Lebenszeit)
EPC	Electronic Power Conditioner
EPL	Eclipse Public License
ESA	European Space Agency
ETX	End Transmission (Ende der Datenübertragung)
FAR	Flight Acceptance Review
FE	Frontend
FIFO	First in, first out (Bearbeitet in der Reihenfolge wie eingegangen.)
FOG	Fiber Optical Gyroscope (Faser-optisches Gyroskop)
FPGA	Field Programmable Gate Array
FSF	Free Software Foundation
GaAs	Galliumarsenid
GCC	GNU Compiler Collection
GDB	GNU Project Debugger
GENIUS	Gps Enhanced NavIgation system for the University of Stuttgart micro satellite
GMFE	Generic Modular Frontend
GNU	GNU's Not Unix (Rekursives Akronym)
GPL	GNU General Public License
GPS	Global Positioning System (Globales Positionsbestimmungssystem)
GSOC	German Space Operations Center (Deutsches Raumfahrt-Kontrollzentrum)
GUI	Graphical User Interface (Graphische Benutzerschnittstelle)
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment (Integrierte Entwicklungsumgebung)
IRS	Institut für Raumfahrtsysteme
KARX	Ka-band Receiver (Ka-band Empfänger)
KATX	Ka-band Transmitter (Ka-band Sender)
KUTX	Ku-band Transmitter (Ku-band Sender)
LAN	Local Area Network
LGPL	GNU Lesser General Public License
LSB	Least Significant Byte (Niederwertigstes Byte)
LTDN	Local Time of Descending Node (Lage des absteigenden Knotens)
MCS	Missionskontrollsystem (Mission Control System)
MDR	Mission Definition Review
MDVE	Model-based Development and Verification Environment (Modell-basierte Entwicklungs- und Verifikationsumgebung)
MGM	Magnetometer (Magnetfeldsensor)
MGT	Magnetorquer (Magnetischer Drehmomenterzeuger)
MIB	Mission Information Base

MICS	Multispectral Image Camera System (Multispektralbild Kamerasystem)
MMI	Man-Machine Interface (Mensch-Maschine Schnittstelle)
MySQL	„my S-Q-L“, Name eines verbreiteten Datenbank Verwaltungssystems
NCTRS	Network Control and Telemetry Routing System
OBC	On-Board Computer
OBC-SIM	On-Board Computer Simulator
OBSW	On-Board Software
OMG	Object Management Group
P/L	Payload (Nutzlast)
PAL	Low Power Amplifier
PAMCAM	Panoramic Camera (Panorama Kamera)
PC	Personal Computer
PCDU	Power Control and Distribution Unit (Energieversorgungseinheit)
PCF	I^2C-Buscontroller
PCI	Peripheral Component Interconnect (Peripheriekomponenten Schnittstelle)
PDR	Preliminary Design Review
PRR	Preliminary Requirements Review
PSS	Power Supply System (Energieversorgungssystem)
PUS	Packet Utilization Standard
QR	Qualification Review
RAID	Redundant Array of Inexpensive Disks (Redundante Anordnung von preiswerten Datenträgern)
RS422	Recommended Standard 422 (Empfohlener Standard 422)
RW	Reaction Wheel (Reaktionsrad)
S&M	Structure and Mechanics (Struktur- und Mechaniksystem)
SCM	Source Configuration Management
SCOE	Special Check-Out Equipment
SCOS	Spacecraft Control and Operation System
SIL	Satellite Integration Laboratory (Satelliten-Integrations-Labor)
SOC	State-of-Charge (Ladezustand)
SQL	Structured Query Language
SRX	S-band Receiver (S-band Empfänger)
SSR	System Specification Review
STF	Simulator-Transfer-Frames
STR	Star Tracker (Sternensensor)
STX	S-band Transmitter (S-band Sender)
STX	Start Transmission (Beginn der Datenübertragung)
SuS	Sun Sensor System (Sonnensensorsystem)
SVF	Software Verification Facility
TC	Telecommand (Telekommando)
TCS	Thermal Control System (Thermalkontrollsystem)

TCP/IP	Transmission Control Protocol/Internet Protocol
TICS	Thermal Infrared Camera System (Thermalinfrarot Kamerasystem)
TM	Telemetry (Telemetrie)
TT&C	Telemetry, Tracking and Command System (Kommunikationssystem)
TWT	Traveling Wave Tube (Wanderfeldröhre)
µASC	micro Advanced Stellar Compass
UHF	Ultra High Frequency
UML	Unified Modeling Language
URL	Uniform Resource Locator
URX	Ultra High Frequency (UHF) Receiver (UHF Empfänger)
USA	United States of America (Vereinigte Staaten von Amerika)
USB	Universal Serial Bus
UTX	Ultra High Frequency (UHF) Transmitter (UHF Sender)
VHF	Very High Frequency (VHF)
VRX	Very High Frequency (VHF) Receiver (VHF Empfänger)
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
XOR	Exclusive or (Exklusives Oder)

0 Einleitung

Das innovative Stuttgarter Kleinsatellitenprogramm des Instituts für Raumfahrtsysteme (IRS) der Universität Stuttgart stellt sich der Aufgabe, Kleinsatelliten zur Technologie-Evaluation und Durchführung von wissenschaftlichen Experimenten zu bauen, zu starten und im Erdorbit zu betreiben. Ziel des Programms ist nicht nur als abgeschlossenes Ergebnis die Kleinsatelliten im Erdorbit, sondern vielmehr durch den gesamten Entwicklungsprozess der Satelliten wissenschaftlichen und technischen Nachwuchs auszubilden und zu qualifizieren. Das Entwicklungsteam besteht zum größten Teil aus wissenschaftlichen Angestellten, die ihr Promotionsvorhaben diesem Thema widmen und die Verantwortung für je eine Fachdisziplin übernommen haben. Sie betreuen studentische Hilfskräfte, Studien- und Diplomarbeiten zu diesem Thema und involvieren damit zusätzlich eine Vielzahl von Studenten der raumfahrttechnischen und wissenschaftlichen Fachrichtung. Zusammenfassend stellt sich der Nutzen des Programms durch wissenschaftliche Beiträge der Experimente im Erdorbit, durch den erfolgreichen Einsatz erprobter, neuartiger Technologien in Satellitensystemen und die Schaffung eines Kompetenzzentrums für den Bau und Betrieb von Kleinsatelliten am Stuttgarter Institut für Raumfahrtsysteme dar. Die am Projekt beteiligten Studenten und Angestellten repräsentieren durch ihre Erfahrungen mit den Kleinsatelliten einen Teil dieses errungenen Know-hows und bieten nach Studienabschluss dem Arbeitsmarkt attraktive Fähigkeiten. Letzteres ist sicherlich ein Grund für die starke Beteiligung der Raumfahrtindustrie am Stuttgarter Kleinsatellitenprogramm.

Bis in die jüngste Vergangenheit waren viele Missionen und Experimente im Erdorbit nur mit großen Satelliten möglich, die zu diesem Zweck mit einer Vielzahl von Systemen ausgestattet waren. Die Entwicklung dieser großen Satelliten erforderte ein erhebliches Budget, große menschliche und technologische Ressourcen und führte nach einer ebenso langen Entwicklungszeit dazu, dass die integrierte Technologie bereits vor dem Start nicht mehr dem aktuellen Stand der Technik entsprach. Die rapide technologische Miniaturisierung ermöglicht heutzutage die Nutzlast-Ausstattung von Kleinsatelliten mit vergleichbarer Funktionalität und Komplexität. Die Kleinsatelliten können gleichzeitig in viel kürzerer Zeit, bei deutlich geringeren Kosten und einem kleineren Team entwickelt werden. Dies hat Kleinsatelliten sowohl für die traditionellen Satellitenhersteller als auch für Universitäten zu einer erschwinglichen Technologie-Evaluierungsplattform werden lassen.

Der erste Satellit in diesem Programm ist der drei-Achsen stabilisierte Kleinsatellit *Flying Laptop*. Er soll in einen 500-900 km hohen sonnen-synchronen Erdorbit starten und neben der Technologie-Demonstration zu Erdbeobachtungszwecken eingesetzt werden. Die vorliegende Dissertation wurde im Rahmen des Stuttgarter Kleinsatellitenprogramms mit der konkreten Anwendung im *Flying Laptop* Projekt durchgeführt.

Die Dissertation ist in thematisch differenzierte Kapitel gegliedert. Die Aufgabenstellung wird im ersten Kapitel in den derzeitigen Stand der Entwicklung von Satelliten und Kleinsatelliten eingeordnet. Es folgen grundlegende Informationen und Erläuterungen zu den behandelten Themen und Basistechnologien. Das dritte Kapitel geht detailliert auf die Durchführung und Umsetzung der Aufgabenstellung ein, worauf im Anschluss die Ergebnisse diskutiert werden. Abschließend formuliert der Ausblick die noch bevorstehenden und durch diese Arbeit ermöglichten weiterführenden Aufgaben.

1 Einordnung der Aufgabenstellung

In der traditionellen Satellitenentwicklung wurden bisher eine Vielzahl von aufwendigen und damit auch kostspieligen Entwicklungsmodellen erstellt. Jede Fachdisziplin, wie zum Beispiel die Lageregelung oder Struktur- und Thermal-Design, benötigte dabei ein spezialisiertes Modell, um die funktionellen Fragestellungen des Systemdesigns im Experiment unter Weltraumbedingungen zu analysieren. Die letztendliche Verifikation des als realisierbar identifizierten Designs erfolgte dann am Flugmodell des Satelliten.

Bereits seit einigen Jahren verwenden renommierte Satellitenhersteller besondere Technologien, um bereits während der Satellitenentwicklung und Fertigung den größtmöglichen Missionserfolg zu gewährleisten. Eine dieser Technologien ist die Modell-basierte Entwicklungs- und Verifikationsumgebung (Model-based Development and Verification Environment, MDVE) [Eickhoff et al. '03] der Firma EADS Astrium GmbH aus Friedrichshafen. Sie bietet ein systematisches und standardisiertes Entwicklungs- und Verifikationsrahmenwerk im Sinne eines systemweiten Satellitensimulators. In diesem Simulator können alle Satellitenkomponenten modelliert und auf funktionaler Ebene verifiziert werden. Auf diese Weise können aufwendige und kostenintensive Entwicklungsmodelle einzelner Subsysteme und Schlüsseltechnologien wegfallen [Eickhoff et al. '07]. Als technologische Voraussetzung bringt gerade diese Entwicklungstechnologie budgetschwache Kleinsatellitenprojekte der Realisierung einen deutlichen Schritt näher und erhöht gleichzeitig die Güte des Systemdesigns des Satelliten. Insbesondere sei hier die Verifizierung der Genauigkeit des Lageregelungssystems angesprochen.

Bei der Entwicklung von Kleinsatelliten kommen bisher deutlich weniger komplexe technische Systeme zum Einsatz. Dies liegt sicherlich auch am weniger komplexen Design der Kleinsatelliten. Die verwendeten Test und Simulationsprinzipien kann man in zwei Gruppen einteilen: Erstens die rein funktionale Simulation, wie sie auch der Autor dieser Dissertation bereits in seiner Diplomarbeit durchgeführt hat ([Falke '04], [Falke et al. '04], [Amini et al. '05], [Curti et al. '07]). Zweitens die detailliertere Simulation unter Berücksichtigung des realen Verhaltens der projektspezifischen Satelliten Hardware mit Teilen eben dieser Hardware im Simulationskreislauf ([Tortora et al. '06], [Zheng, Low '06]). Allen beiden Gruppen ist in ihren bisherigen Anwendungsfällen gemeinsam, dass sie nur einzelne Subsysteme bzw. Fragestellungen umfassen und keine darüber hinausgehenden Tests des Gesamtsystems durchführen können. In diesen einfachen Simulatoren ist es zwar möglich mittels spezieller Geräte, Satelliten Hardware in den Simulationskreislauf einzubinden, aber erst wenn der On-board Computer mit der realen On-board Software verwendet wird [Enderle et al. '04], bietet der Testaufbau erweiterte Verifikationsmöglichkeiten. Doch leider wird dieses Testpotential in Kleinsatellitenprojekten bisher noch nicht voll ausgeschöpft, da der Testaufbau auf einzelne Subsysteme begrenzt ist. Die systemweite Betrachtung eines Raumfahrzeugs in Simulationen und Tests steigert jedoch die Zuverlässigkeit des Systems im Ganzen. Dies wird in Kleinsatellitenprojekten aus finanziellen und zeitlichen Gründen meist nicht durchgeführt und somit ist die Verifikation der einstmals gestellten Anforderungen an den Kleinsatelliten als Gesamtsystem nicht bzw. nur eingeschränkt möglich.

Nachdem diese gravierende technische Lücke und das daraus resultierende Potential am Institut für Raumfahrtsysteme erkannt wurde, ist es nun Ziel dieser Dissertation in Kooperation mit EADS Astrium GmbH diese Entwicklungsplattform erstmalig für die Anwendung in universitären Kleinsatellitenprojekten nutzbar zu machen und im Speziellen die FPGA-basierte (Field Programmable Gate Array) On-board Computer Architektur des *Flying Laptop* in dieses Simulationssystem zu integrieren. Dies bezieht sich auf die Software-Verifikations-Einrichtung (Software Verification Facility, SVF), die basierend auf einem Linux-Derivat am Institut für Raumfahrtsysteme erstellt werden soll.

Ziel dieser Dissertation ist es im Weiteren auch, den Systemsimulator zu einer Standardanwendung für die Kleinsatellitenprojekte im Stuttgarter Kleinsatelliten Programm zu entwickeln, denn heutzutage gehört diese systemweite Simulationstechnologie zum Standard in der Satellitenentwicklung. Es ist folglich ein wichtiger und logischer Schritt, diese Technologie auch in Kleinsatellitenprojekten anzuwenden. Der hohe finanzielle Aufwand für die Computer Hardware und die benötigten kommerziellen Software Lizenzen wurde dafür vom Institut für Raumfahrtsysteme bereitgestellt. Zur Eingrenzung und langfristigen Kostenkontrolle lag ein zusätzliches Augenmerk dieser Dissertation auf der Anwendung von quell-offener Software im gesamten Entwicklungs- und Anwendungsbereich des Systemsimulators. Dies ist beim derzeitigen Trend der weltweiten Softwareentwicklung hin zu quell-offenen, aber dennoch professionellen Softwarewerkzeugen ein durchaus realisierbares Unterfangen.

Der konkrete Gewinn aus der Thematik der Systemsimulation für das *Flying Laptop* Projekt lässt sich in zwei Bereiche aufteilen: Erstens unterstützt und ermöglicht die Software-Verifikations-Einrichtung des Systemsimulators Simulationen des gesamten Satelliten unter Echtzeitbedingungen [Falke et al. '06]. Dies ist besonders für die Entwicklung der On-board Kontrollalgorithmen, der Lageregelungsalgorithmen, dem Test von Übergängen der Betriebsmodi des Satelliten und kompletter Flugprozeduren von großer Bedeutung. Der zweite große Gewinn ist das Werkzeug zur Flughardware-Verifizierung selbst. Angefangen mit der Software-Verifikations-Einrichtung bis hin zu einem „FlatSat“ Teststand kann die Echtzeit-Simulatorumgebung des MDVE als Teststand für die gesamte Satellitenhardware durch sukzessives Integrieren der Komponenten in den Simulationskreislauf genutzt werden. Die schrittweise verbesserte und verifizierte Teststand Funktionalität führt schließlich zu einem ernst zu nehmenden Werkzeug, dass während des gesamten Flughardware-Verifikationsprozesses Anwendung finden kann.

2 Grundlagen und Basistechnologien

Dieses Kapitel behandelt die theoretischen Grundlagen und den anwendungsorientierten Hintergrund dieser Dissertation. Auf das Stuttgarter Kleinsatelliten Programm wird hier am Beispiel des *Flying Laptop* Projekts näher eingegangen. Es folgen Erläuterungen zur Theorie der Simulation und eine ausführliche Einweisung in die verwendete Modell-basierte Entwicklungs- und Verifikationsumgebung der Firma EADS Astrium GmbH. Das abschließende Unterkapitel vermittelt schließlich die softwaretechnischen Voraussetzungen in diesem Zusammenhang.

2.1 Der Kleinsatellit *Flying Laptop* im Rahmen des Stuttgarter Kleinsatelliten Programms

Im Jahr 2002 wurde das „Stuttgarter Kleinsatellitenprogramm“ am Institut für Raumfahrtsysteme (IRS) der Universität Stuttgart ins Leben gerufen. Zu den Zielen des Programms gehört es einerseits, Studenten die Möglichkeit theoretischer und praktischer Erfahrungen in einem solchen Projekt zu bieten. Andererseits wird das Programm zeigen, dass mit der aktuellen Technologie ein Universitätsinstitut in der Lage ist, einen Kleinsatelliten zu bauen, zu starten und zu betreiben. Dazu wird am Institut für Raumfahrtsysteme die Infrastruktur für den Bau und Betrieb des Satelliten geschaffen. Im Rahmen des Kleinsatellitenprogramms sind vier Kleinsatelliten mit unterschiedlichen Zielsetzungen geplant:

- Der Erdbeobachtungssatellit *Flying Laptop* wird als erstes entwickelt und ist mit der Technologiedemonstration als einem seiner Ziele ein wichtiges Element zur Vorbereitung der nachfolgenden Missionen. Wissenschaftliche Herausforderung ist die Vermessung der Bi-direktionalen Reflektanzverteilungsfunktion (BRDF) mit optischen Kameras.
- Basierend auf den Basissystemen des Vorgängers, aber ausgestattet mit ebenfalls am IRS entwickelten elektrischen Antriebssystemen wird der Kleinsatellit *Perseus* erste selbstständige Erfahrungen mit angetriebenen Raumflugkörpern ermöglichen. Weiterhin dient ein Teleskop der Erforschung des Weltraums im ultravioletten Strahlungsbereich.
- Unter dem Arbeitstitel *„Lunar Mission BW1“* wird eine Kleinsatellitenmission vorbereitet, die mit den bereits bei *Perseus* getesteten elektrischen Antriebssystemen ausgestattet sein wird. Ihr Missionsziel ist es, mit diesen Antriebssystemen eine kreisförmige Mondumlaufbahn zu erreichen und dort für sechs Monate wissenschaftliche Untersuchungen durchzuführen.
- Das Wiedereintrittsfahrzeug *Cermit/ Desire* soll ähnlich einem Kleinsatellitenprojekt mit Kooperationspartnern aus dem universitären und industriellen Bereich im Rahmen eines Sonderforschungsbereiches entwickelt und betrieben werden. Ziel ist die Erprobung von neuartigen Technologien und Lenkkonzepten für Wiedereintrittsfahrzeuge.

2.1.1 Der erste Stuttgarter Kleinsatellit: *Flying Laptop*

Der Kleinsatellit *Flying Laptop* (siehe Abbildung 2.1) ist der erste am Institut für Raumfahrtsysteme gebaute Satellit. Er besitzt einen quaderförmigen Körper mit einer Kantenlänge von durchschnittlich 60 cm bei einer angestrebten Gesamtmasse von weniger als 120 kg. Sein primäres Missionsziel ist die Technologie Demonstration sowohl für das Satellitenkonzept selbst, als auch für den Rest des Stuttgarter Kleinsatellitenprogramms und insbesondere die geplante Mondmission *„Lunar Mission BW1“*. Zielorbit des Kleinsatelliten ist ein polarer, sonnen-synchroner niedriger Erdorbit in 500-900 km Höhe.

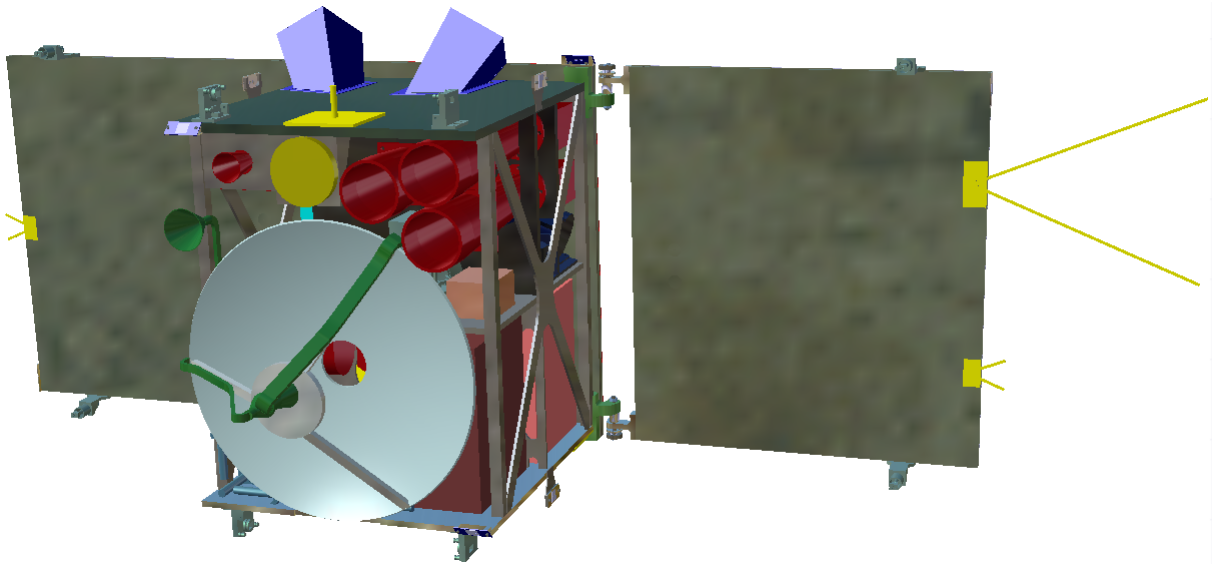


Abbildung 2.1: Modellzeichnung des Kleinsatelliten Flying Laptop

Während die Kommunikation mit einem Satelliten in einem niedrigen Erdorbit auch für eine Universität einfach zu realisieren ist, wird diese Aufgabe bei einer größeren Distanz zum Satelliten, wie zum Beispiel in einem Mondorbit, ungleich komplexer. Die traditionelle Niedrigfrequenz-Kommunikation würde eine Parabolantenne mit 30 m Durchmesser erfordern, die für einen Universität finanziell nicht tragbar ist. In Folge dessen wird die Mondmission *„Lunar Mission BW1“* höhere Kommunikationsfrequenzen im sogenannten Ka-Band nutzen, die bei gleicher Empfangsqualität nur eine Parabolantenne mit 3 m Durchmesser benötigen. Als Vorgängermission wird bereits der *Flying Laptop* diese Technologie anwenden und erproben. Er ist mit einer Parabolantenne im Cassegrain System von 50 cm Durchmesser ausgestattet und wird die Ka-Band Frequenz für den Datentransfer zur Bodenstation mit hoher Geschwindigkeit nutzen. Das Cassegrain Spiegelsystem wird auch als optisches System für die Kamera im thermischen Infrarot verwendet.

Der On-board Computer des *Flying Laptop* wird aus einem Field Programmable Gate Array (FPGA) basierten System bestehen. Gegenüber normalen Mikroprozessoren mit fester interner Logik kann die interne Logik des FPGA re-programmiert werden, um die verbesserten bzw. erweiterten Funktionalitäten des On-board Computers abzubilden. Die Anwendung des FPGA-basierten Computers führt durch die hohe Parallelität der On-board Kontrollalgorithmen gegenüber einer konventionellen Prozessorarchitektur zu einer signifikant gesteigerten Leitung.

Das präzise Lageregelungssystem besteht ebenfalls aus neu entwickelten Komponenten: Das Magnetometer, das Faser-optische Kreiselsystem und die verbesserte Version eines Sternensensors. Diese Komponenten und der leistungsstarke On-board Computer ermöglichen nicht nur eine Ausrichtungsgenauigkeit von 2,5 Bogensekunden, sondern auch eine hoch agile Manövrierfähigkeit.

Neben der Technologie-Demonstration werden auch Erdbeobachtungsexperimente durchgeführt, wobei die Messung der Bi-direktionalen Reflektanzverteilungsfunktion eine große Rolle spielt. Die BRDF beschreibt die Abhängigkeit der Reflektion vom Winkel des Betrachters und der Sonneneinstrahlung. Zusätzlich werden Bodenmessungen in 5 m Höhe und Messungen aus 300 m Höhe gemacht, um die Datenbasis dieser Abhängigkeitsbeziehung zu erweitern. Der *Flying Laptop* ist zu diesem Zweck mit drei Kameras in den Spektralbereichen des visuellen und nahen Infrarot ausgestattet. Alle Kameras haben eine vergleichbare geometrische Auflösung.

Eine neue Methode der Niederschlagsmessung wird ebenfalls getestet. Physikalische Grundlage der Messung ist die unterschiedliche Strahlungsabschwächung im Ka- und Ku-Band, die proportional der Niederschlagsrate ist. Dadurch kann die regionale Niederschlagsrate direkt bestimmt werden. Weiterhin wird die Ka-Band Antenne für Messungen der atmosphärischen Strahlungsabschwächung im Ka-Band verwendet. Die bisher für den Kontinent Europa existierende Datenmenge ist noch viel zu gering und der *Flying Laptop* wird helfen diese Datenbasis zu erweitern.

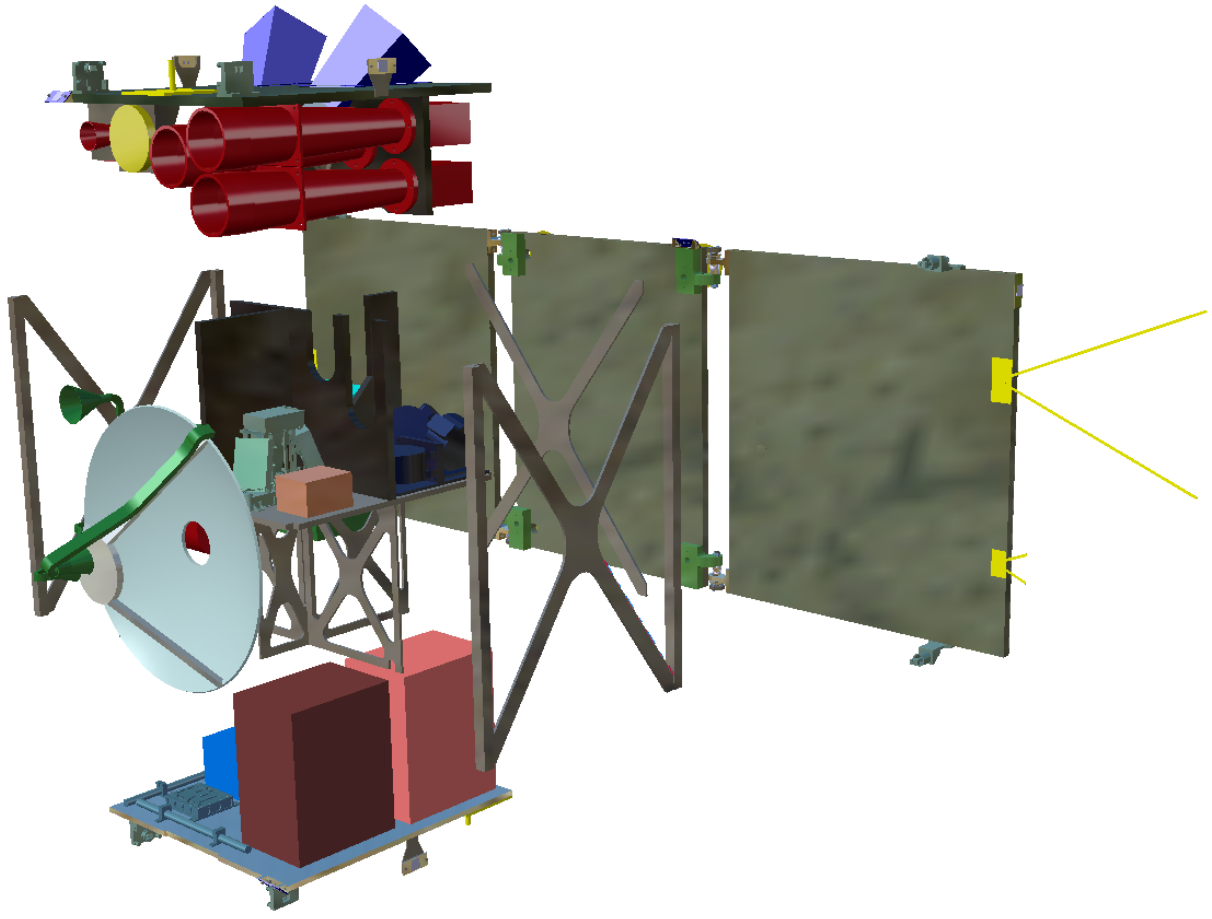


Abbildung 2.2: Baugruppen und Module des Kleinsatelliten Flying Laptop

Strukturell ist der *Flying Laptop* in drei Module aufgeteilt (siehe Abbildung 2.2): Das Nutzlastmodul, das Kernmodul und das Servicemodul. Jedes Modul besteht aus einer Basisplatte auf der die jeweiligen Komponenten montiert sind. Querträger und Zwischenwände sorgen für die Stabilität der Struktur. Wegen der guten thermischen Leitfähigkeit und einfachen Herstellung sind das Kernmodul, das Servicemodul und die Querträger aus Aluminium gefertigt. Das Nutzlastmodul und die optische Bank sind aufgrund der hohen thermischen Formstabilität aus Kohlefaser-verstärktem Kunststoff gefertigt.

Das Nutzlastmodul beherbergt die Kamerasysteme und wissenschaftlichen Messinstrumente des Satelliten. Auf der optischen Bank sind aufgrund der thermischen Formstabilität auch die Sternensensoren montiert. Ein großer Wärmeradiator an der Außenseite des Nutzlastmoduls sorgt für die passive Temperierung der Komponenten. Die wesentlichen Basissysteme des Satelliten sind im Kernmodul und im Servicemodul untergebracht. Der Startadapter für die Montage auf der Rakete ist an der Außenseite des Servicemoduls montiert. Die Geräte mit der höchsten Wärmedissipation sind im Servicemodul direkt an einem weiteren Wärmeradiator im Startadapter montiert. Zusätzlich transportiert ein Wärmerohr Wärme vom Kernmodul ebenfalls zum Radiator im Nutzlastmodul. Die große Parabolantenne ist parallel der optischen Achse des Kamerasystems am Kernmodul befestigt. Drei Solarpaneele sind an den Außenflächen des Satelliten befestigt, die beiden äußeren werden nach dem Start im Orbit aufgeklappt, so dass alle Paneele eine komplanare Ebene bilden.

2.1.2 Systemkomponenten des *Flying Laptop*

In der folgenden Abbildung 2.3 sind alle Komponenten des Kleinsatelliten *Flying Laptop* dargestellt und den jeweiligen Subsystemen zugeordnet. Weiterhin sind alle elektrischen Schnittstellen und Kommunikationsverbindungen der Komponenten untereinander eingezeichnet. In der Regel besitzt jede Komponente eine elektrische Verbindung zum Energieversorgungssystem und eine Kommunikationsverbindung zum FPGA-basierten On-board Computer. Im Folgenden werden die einzelnen Komponenten etwas genauer erläutert, um das Verständnis der späteren Modellentwicklung der jeweiligen Komponente im Systemsimulator zu unterstützen.

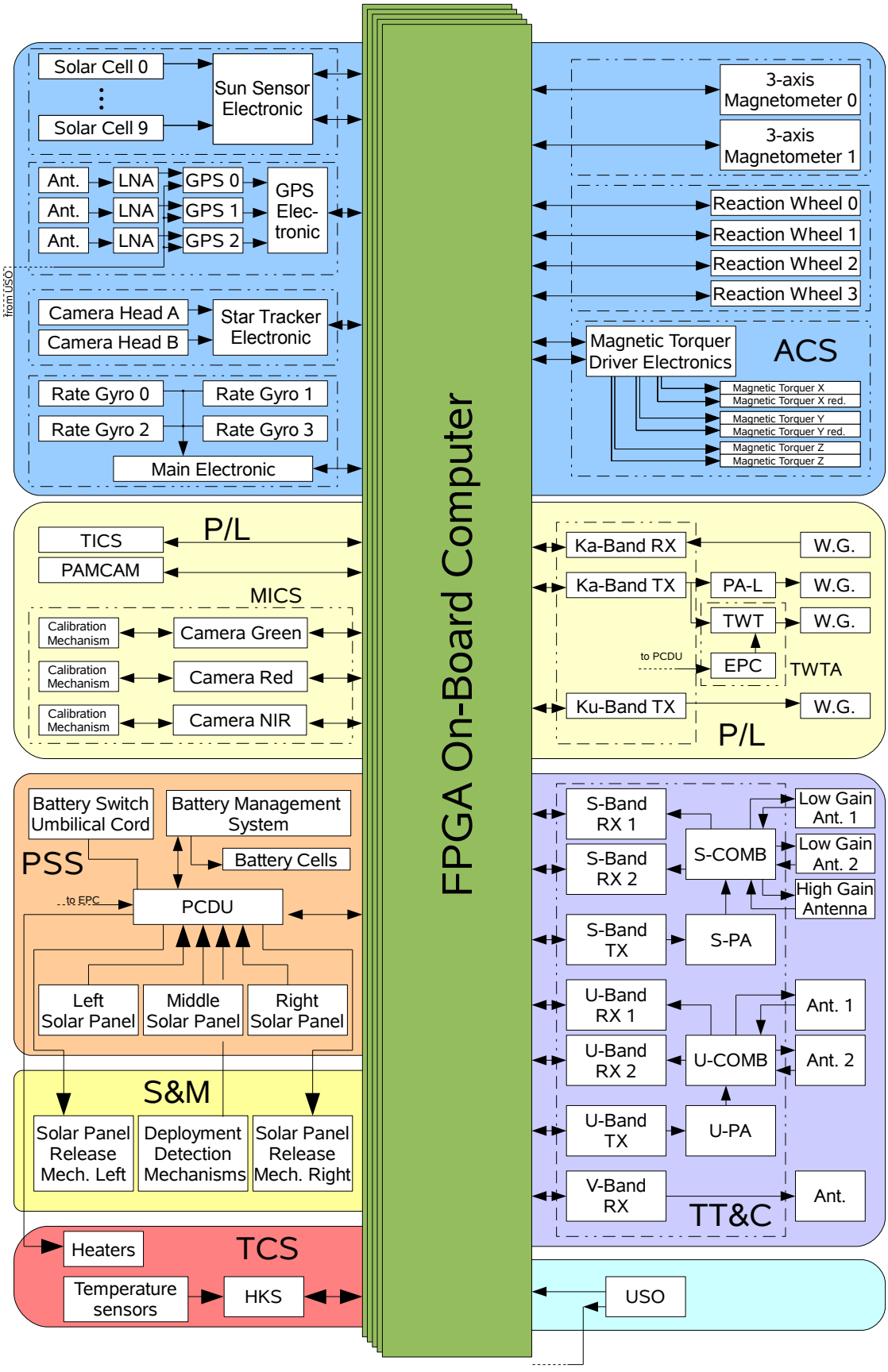


Abbildung 2.3: Übersicht der Systemkomponenten, der elektrischen Schnittstellen und Kommunikationsverbindungen des Satelliten Flying Laptop

2.1.2.1 Das Lageregelungssystem (Attitude Control System, ACS)

Die Drehratensensoren (Fiber Optical Gyroscope, FOG)

Zur Messung des aktuellen Bewegungszustandes des Satelliten wird ein System von Drehratensensoren verwendet. Das System besteht aus vier baugleichen faseroptischen Gyroskopen, die aus Gründen der Redundanz in Tetraeder-Anordnung im Satelliten montiert sind. Die Tetraeder-Anordnung ermöglicht auch bei Ausfall eines Sensors weiterhin die Messung der Drehraten in allen drei Achsen. Die Sensoren haben eine Drift-Rate von ca. 3° pro Stunde und sind in einem strahlungsgeschützten Gehäuse untergebracht.

Das GPS System (GPS System, GPS)

Der *Flying Laptop* ist mit insgesamt drei eigenständigen Phoenix GPS Empfängern ausgestattet, deren Antennen in L-Anordnung in drei Ecken auf dem mittleren Solarpaneel montiert sind. Die Intention dieser Anordnung ist das Experiment GENIUS (Gps Enhanced NavIgation system for the University of Stuttgart microsatellite), dessen Ziel eine Lagebestimmung des Satelliten aufgrund der Phasenverschiebung des GPS-Signals ist. GENIUS wird in Kooperation mit dem German Space Operations Center (GSOC, DLR) geplant und durchgeführt. Jede der drei Empfängereinheiten ist separat aktivier- und ansprechbar. Die Empfängereinheiten und die zugehörige Elektronik sind im Inneren des Satelliten untergebracht. Für die Nutzung im Satelliten bieten die GPS-Empfänger die Positions- und Geschwindigkeitsbestimmung und Zeitinformationen in Echtzeit mit einer Genauigkeit von 10 m, 0,1 m/s und 1 µs an. [Grillmayer et al. '05b]

Der Sternensensor (Star Tracker, STR)

Der Sternensensor micro Advanced Stellar Compass (µASC) der Technischen Universität Dänemark bietet eine Lagebestimmung innerhalb eines Bereichs von 2 Bogensekunden. Nachdem der Satellit stabilisiert ist und mit einer maximalen Drehrate von 1,2° pro Sekunde rotiert, liefert der Sternensensor regelmäßig hochgenaue Lageinformationen. Zum Erreichen der vollen Genauigkeit in allen Achsen und um Blendungen des Sensorkopfes während Drehmanövern zu verhindern, ist ein zweiter Sensorkopf mit um 22,5° gedrehter optischer Achse im Satelliten montiert. Die Datenprozessierungseinheit (Data Processing Unit, DPU) empfängt die Bilder beider Sensorköpfe und fasst diese zu einer hochgenauen Lageinformation zusammen.

Das Sonnensensorsystem (Sun Sensor System, SuS)

Insgesamt 16 kleine Solarzellen an 8 verschiedenen Orten montiert, bilden das Sonnensensorsystem des *Flying Laptop*. Je zwei zueinander redundante Solarzellen pro Montageort decken ihren entsprechenden Halbraum ab. Zwei unabhängige Analog/Digitalwandler transformieren die generierte Spannung an den Solarzellen von je 8 unterschiedlichen Positionen in ein digitales Signal. Über je einen I^2C-Datenübertragungsbus können diese Informationen vom On-board Computer abgefragt werden. In den On-board Kontrollalgorithmen wird aus den übertragenen Informationen über die generierte Spannung an dem Widerstand bekannter Größe der generierte Solarzellensensorstrom bestimmt und anschließend der aktuelle Vektor zur Sonne berechnet.

Die Magnetfeldsensoren (Magnetometer, MGM)

Zur Messung des Erdmagnetfeldes sind insgesamt zwei drei-Achsen Magnetfeldsensoren von ZARM im Satelliten montiert. Die Sensoren sind über eine digitale Schnittstelle ansprechbar.

Die magnetischen Drehmomenterzeuger, Magnetorquer (Magnetic Torquer, MGT)

Zur Lagestabilisierung im `SAFE` Mode, zur langsamen Lageänderung und zur Entsättigung der Reaktionsräder werden die magnetischen Drehmomenterzeuger verwendet. Sie bestehen aus einer Kupferspule, die bei Stromdurchfluss ein Magnetfeld erzeugt, welches sich gegenüber dem Erdmagnetfeld abstützt und somit ein Drehmoment erzeugt. Zur Drehmomenterzeugung in allen drei Achsen sind insgesamt drei Magnetorquer orthogonal zueinander im Satelliten montiert. Die Magnetorquer besitzen ein lineares Dipolmoment von 6 Am^2. Die zugehörige Elektronik erlaubt über eine digitale I^2C-Schnittstelle das Ein- bzw. Ausschalten des Stromes und die Vorgabe der Stromrichtung durch die Kupferspule.

Die Reaktionsräder (Reaction Wheel, RW)

Zur schnellen Lageänderung werden die Reaktionsräder verwendet. In einer Tetraeder-Anordnung sind insgesamt vier Reaktionsräder montiert und erlauben auch bei Ausfall eines Rades die Drehmomenterzeugung in allen drei Achsen. Jedes Reaktionsrad hat eine Drehmomentkapazität von 0,12 Nms und einen Reaktionsmoment von 5 mNm über einen Drehzahlbereich von ±3000 Umdrehungen pro Minute. Das akkumulierte Drehmoment der Reaktionsräder wird über die Magnetorquer entsättigt, so dass die Drehzahl der Reaktionsräder niedrig gehalten werden kann.

2.1.2.2 Das Kommando- und Datenverarbeitungssystem (Command and Data Handling System, C&DH)

Der On-board Computer

Beim *Flying Laptop* besteht das Kommando- und Datenverarbeitungssystem allein aus dem On-board Computer. Dieser basiert auf FPGA Technologie und ist rekonfigurierbar, redundant und selbst-kontrollierend mit einer hohen Rechenleistung. Das On-board Computer System besteht aus vier Zentralen Prozessierungsknoten (Central Processing Nodes, CPN) und einer Kommando Dekodier- und Entscheidungseinheit (Command Decoder and Voter, CDV). Die fünf Komponenten sind über ein Schnittstellenboard untereinander und mit der restlichen Satellitenhardware verbunden. Jeder der zentralen Prozessierungsknoten und auch die separate Kommando Dekodier- und Entscheidungseinheit bestehen aus einem rekonfigurierbarem FPGA-basiertem Computer. Insgesamt bildet das On-board Computer System ein konservatives Design, da aus den vier CPN's ein Hauptknoten identifiziert wird. Dies beruht auf der Annahme, dass jeder zentrale Prozessierungsknoten die gleichen Aufgaben in der gleichen Zeit löst und dementsprechend alle Input/Output Signale identisch sind. Der CDV identifiziert anschließend aus der größten Menge der Übereinstimmungen einen Hauptknoten und leitet dessen Signale nach außen zu den Peripheriekomponenten weiter. Alle Peripheriekomponenten sind parallel zueinander angeschlossen, in der Summe mehr als 200 Datenleitungen. [Kuwahara et al. '07]

2.1.2.3 Das Energieversorgungssystem (Power Supply System, PSS)

Die Solarpaneele

Auf drei separaten Solarpaneelen sind insgesamt 274 GaAs (Galliumarsenid) Solarzellen aufgeklebt. Die Anordnung der Solarpaneele ist flügelformig zu beiden Seiten des Satellitenkubus. Nach dem Start werden die beiden äußeren Paneele über einen durch Federkraft unterstützten Entfaltungsmechanismus ausgebreitet und rasten in ihrer Endstellung ein. Auf den äußeren Solarpaneelen sind je 15 Solarzellen zu je 7 Strings zusammengefasst, während auf dem mittleren Solarpaneel 4 Strings mit je 16 Solarzellen verklebt sind. Die eine zusätzliche Solarzelle je String auf dem mittleren Solarpaneel kompensiert die höhere Degradation der dort verklebten Solarzellen aufgrund der höheren Temperatur am Satellitenkubus. Jede einzelne Solarzelle kann im Fall eines Defekts durch eine Überbrückungsdiode umgangen werden. Die Strings jedes Solarpaneels werden zu je einer Versorgungsleitung zusammengefasst.

Die Batterie

Aus insgesamt sechs in Serie geschalteten Batteriezellen besteht das Batteriesystem des *Flying Laptops*. Aufgrund der wesentlich höheren Energiedichte sind Lithium-Ionen Batteriezellen mit einer Kapazität von 50 Ah (Begin-of-life, BOL) bei einer Nennspannung von 4,1 V ausgewählt worden. Jede Batteriezelle kann im Falle eines Defekts durch eine Überbrückungsdiode umgangen werden. Das Batteriemanagementsystem beinhaltet ein Cell-Balancing System und stellt die ordnungsgemäße Batterieladung bei konstantem Strom bzw. anschließend bei konstanter Ladeschlussspannung sicher. Weiterhin schützt es die Batterie vor Überladung bzw. vor Über- und Unterspannung und überwacht die Zellspannungen bzw. Zelltemperaturen. Die Temperaturkontrolle der Batterie wird über redundante Heizelemente realisiert, die über Bimetall-Streifen aktiviert bzw. deaktiviert werden. Das Batteriemanagementsystem ist Teil der Energiekontroll- und Verteilungseinheit, die im Folgenden erläutert wird.

Energiekontroll- und Verteilungseinheit (Power Control and Distribution Unit, PCDU)

Die Energiekontroll- und Verteilungseinheit beherbergt das Batteriemanagementsystem, welches die Ladung und Entladung der Batterie überwacht. Weiterhin versorgt die PCDU die Peripheriekomponenten des *Flying Laptop* mit geregelten Busspannungen in unterschiedlichen Niveaus. Die einzelnen Spannungsniveaus werden von Konvertern basierend auf der Batteriespannung erzeugt. Aus Gründen der Redundanz sind die Peripheriekomponenten auf die beiden Konverter so aufgeteilt, dass bei Ausfall eines Konverters eine Restfunktionalität erhalten bleibt. Die Konverter sind so ausgelegt, dass sie auch bei Ausfall einer Batteriezelle und dem damit verbundenen Spannungsabfall arbeiten können. Jede Versorgungsleitung zu den Peripheriekomponenten ist durch eine Strombegrenzungssicherung gegen zu hohen Stromverbrauch aufgrund einer Fehlfunktion der Komponente gesichert.

2.1.2.4 Das Kommunikationssystem (Telemetry, Tracking and Command System, TT&C)

Die UHF, VHF und S-band Kommunikationsgeräte

Das Kommunikationssystem besteht aus mehreren Kommunikationsgeräten in unterschiedlichen Frequenzbereichen. Zum Zeitpunkt der Anfertigung dieser Dissertation sind diese Geräte nur mit ihren wichtigsten Charakteristiken definiert. Die Elektronikplatinen aller Geräte sollen in einem Gehäuse, der Kommunikationsbox untergebracht werden. Der *Flying Laptop* besitzt zwei UHF Empfänger (URX), einen UHF Sender (UTX) und einen VHF Empfänger (VRX). Weiterhin sind zwei S-band Empfänger (SRX) und ein S-band Sender (STX) vorhanden.

2.1.2.5 Das Struktur- und Mechaniksystem (Structure and Mechanics, S&M)

Die Gesamtstruktur des *Flying Laptop* ist zur einfacheren Montage in drei Module unterteilt: Das Core Modul, das Service Modul und das Nutzlast Modul (siehe Abbildung 2.2 auf Seite 6). Die drei Solarpaneele sind außen an den Modulen montiert. Die äußeren Paneele werden durch einen Niederhaltemechanismus am Satellitenkubus gehalten und können im Erdorbit durch einen Federkraft-unterstützten Entfaltungsmechanismus entfaltet werden. Sie rasten dann in ihrer Endstellung ein. Der Niederhaltemechanismus basiert auf einer Schmelzschnur, die im Erdorbit unter Stromdurchfluss kleiner Heizer zum Schmelzen gebracht wird. Ist die Schmelzschnur durchgeschmolzen kann die Federkraft des Entfaltungsmechanismus ihre Kraft freisetzen und die Solarpaneele nach außen drücken. Die Aktivierung des Schmelzschnurstromes erfolgt über zwei redundante Stromleitungen in der PCDU, die direkt von der Batteriespannung aus betrieben werden.

Zur Montage der optischen Geräte wird im Payload Modul eine optische Bank aus einer kohlefaserverstärkten Sandwichstruktur verwendet. Die thermischen Eigenschaften der Kohlefaser ermöglichen eine möglichst geringe Wärmeausdehnung und gewährleisten dadurch eine hohe thermische Formstabilität.

2.1.2.6 Das Thermalkontrollsystem (Thermal Control System, TCS)

Die Überwachungsbox (Housekeeping, HK)

In der Housekeeping Box laufen die Leitungen von allen im Inneren oder außen am Satelliten montierten Thermalsensoren zusammen. Die Sensorwiderstände werden zu digitalen Daten umgewandelt und können anschließend über eine I^2C-Datenleitung vom On-board Computer abgefragt werden.

Die Heizelemente

Jedes der drei Strukturmodule des *Flying Laptops*, die Batterie und das thermische Infrarotkamerasystem ist mit zwei redundanten Heizelementen ausgestattet. Die Heizelemente werden durch Bimetall-Streifen aktiviert bzw. deaktiviert und bestehen aus einfachen Widerstandsheizmatten. Sie sind direkt an die Batteriespannung angekoppelt und operieren dementsprechend mit variabler Leistungsspanne.

2.1.2.7 Die wissenschaftliche Nutzlast (Payload, P/L)

Die Ka- und Ku-band Kommunikationsgeräte

Zusätzlich zu den bereits erwähnten Kommunikationsgeräten besitzt der *Flying Laptop* im Ka- und Ku-band zu experimentellen Zwecken weitere Kommunikationsgeräte. Sie bestehen aus einem Ka-band Empfänger (KARX), einem Ka-band Sender (KATX) mit der entsprechenden Wanderfeldröhre (Traveling Wave Tube, TWT) und einem Ku-band Sender (KUTX) mit einem Niedrigenergieverstärker (PAL). Für die Bereitstellung der hohen Spannung für die TWT ist nicht die PCDU, sondern eine Energiekonditionierungseinheit (Electronic Power Conditioner, EPC) verantwortlich. Als Antenne dient eine große Parabolschüssel.

Das Multispektralbild Kamerasystem (Multispectral Image Camera System, MICS)

Für die Bildgebung im sichtbaren und nahen Infrarot ist das MICS Kamerasystem zuständig. Es besteht aus drei Kameras in den Frequenzbereichen grün (530–580 nm), rot (620–670 nm) und nahem Infrarot (830–890 nm)

Die Panorama Kamera (Panoramic Camera, PAMCAM)

Die einzige nicht wissenschaftlichen Zwecken dienende Kamera ist die Panorama Kamera mit einem vergleichsweise großen Bildfeld. Sie wird für publikumswirksame Bilder verwendet und hat einen speziellen Videomodus.

Das Thermalinfrarot Kamerasystem (Thermal Infrared Camera System, TICS)

Ein Cassegrain Spiegelsystem, welches auch für die Ka- und Ku-band Sender verwendet wird, dient der Konzentrierung der Strahlung im thermischen Infrarot, die dann ins Innere das Satelliten zum TICS Kamerasystem geleitet wird. Der Sensor des TICS Kamerasystems ist ein ungekühlter Microbolometer.

2.2 Theorie der Simulation

Die Betrachtung der Simulationstheorie ist ein erster Schritt zur Realisierung des Projekts eine Systemsimulation für einen Kleinsatelliten durchzuführen. Die Autoren Law, Kelton und Zeigler vermitteln in ihren Büchern *Simulation Modelling and Analysis* [Law & Kelton '00] und *Theory of Modelling and Simulation* [Zeigler '76] einen detaillierten Einblick in die Theorie der Simulation und bieten einen Leitfaden mit Ansätzen und Techniken zur Erstellung von Simulationsmodellen. Die folgenden Abschnitte sollen einen groben Überblick der Thematik präsentieren, wie er zum Verständnis in diesem Zusammenhang benötigt wird.

2.2.1 Der Begriff Simulation

Simulation bezeichnet allgemein das Nachahmen des Verhaltens eines Systems oder Prozesses zum Zwecke der Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu kompliziert sind. Insbesondere bei dynamischem Systemverhalten ist Simulation als Werkzeug sehr hilfreich bzw. die einzige Möglichkeit der Systemanalyse.

Konkret bedeutet Simulation, Experimente an einem Modell durchzuführen, um Erkenntnisse über das reale System zu gewinnen. Im Zusammenhang mit Simulation spricht man von dem zu simulierenden System und von einem Simulator als Implementierung oder Realisierung eines Simulationsmodells. Letzteres stellt eine Abstraktion des zu simulierenden Systems mit den Schwerpunkten Struktur, Funktion und Verhalten dar. Der erste Schritt besteht daher aus der Modellfindung bzw. der Modellierung. Die Durchführung einer Simulation an einem Modell mit konkreten Werten ist dann das Simulationsexperiment. Die Ergebnisse werden anschließend interpretiert und auf das reale System zurück übertragen. Variiert man die Ausgangsparameter der Simulation hinsichtlich der Ist-Situation oder einer gewünschten Zielsituation, so können durch den Simulationsverlauf ebenfalls Rückschlüsse auf das reale System gezogen werden. Heutzutage ist mit Simulation fast ausschließlich die Computersimulation gemeint, auch wenn ohne Computer ebenfalls simuliert werden kann.

Um eine wissenschaftliche Untersuchung des Systems durchzuführen werden Annahmen des realen Verhaltens in das Systemmodell eingebaut. Diese Annahmen sind in der Regel mathematische oder logische Beziehungen und erst die Summe aller Annahmen bildet das Gesamtsystem. Wenn die Beziehungen des Modells einfach genug sind, reichen möglicherweise mathematische oder theoretische Methoden aus, um Aussagen über spezielle Fragestellungen zu treffen. Dies ist dann eine analytische Lösung. Die meisten Systeme in der realen Welt sind allerdings für analytische Lösungen zu komplex und erfordern numerische Methoden. In diesem Fall werden Computer zur Lösung der numerischen Modelle eingesetzt. Hohe Bedeutung muss dabei der korrekten Modellierung beigemessen werden, da fehlerhafte oder unzureichende Modellbeziehungen erfahrungsgemäß schnell zu Fehlern führen können.

Per Definition ist ein System eine Menge von Objekten, die nicht unbedingt vom gleichen Typ sein müssen, aber miteinander in logischem Zusammenhang interagieren. Die Modellbeziehungen sind derart geknüpft, dass sie Aussagen über die spezielle Fragestellung der Untersuchung zulassen. Der Systemstatus ist definiert im Bezug auf das Untersuchungsziel als die Menge der Modellparameter, die zur Beschreibung des Systems zu einem bestimmten Zeitpunkt benötigt werden.

2.2.2 Typen von Simulationen und Zeitfortschrittsmechanismen

Systeme können in zwei Gruppen kategorisiert werden: diskrete und kontinuierliche Systeme. In einem diskreten System verändern sich die Parameter nur zu bestimmten (diskreten) Zeitpunkten, wohingegen sich die Parameter in einem kontinuierlichem System kontinuierlich mit der Zeit ändern. Nur wenige Systeme sind vollkommen diskret oder vollkommen kontinuierlich, aber da meist eine Art der Parameteränderung überwiegt, können sie entsprechend klassifiziert werden.

Kontinuierliche Simulation

In der kontinuierlichen Simulation ändern die Zustandsparameter des Modells ihre Werte kontinuierlich mit dem Zeitfortschritt. Die Beziehungen für die zeitliche Wertänderung reichen von einfachen bis zu Differentialgleichungen. Für die meisten kontinuierlichen Systeme sind analytische Lösungen nicht möglich und es muss auf numerische Methoden zurückgegriffen werden. Im letzteren Fall muss allerdings ein konsistenter Anfangszustand gegeben sein, um das Gleichungssystem zu lösen.

Simulation mit diskreten Ereignissen

In der Simulation mit diskreten Ereignissen ändern die Zustandsparameter ihre Werte zu diskreten Zeitpunkten, die im Modell definiert sein müssen. Zu diesen Zeitpunkten können weitere definierte Ereignisse auftreten, die wiederum den Systemzustand verändern oder Aktionen hervorrufen können. Jedes Ereignis ändert entweder den Systemzustand oder aber beeinflusst sein zukünftiges Verhalten.

Kombinierte diskret-kontinuierliche Simulation

Modelle deren Beziehungen sowohl diskrete als auch kontinuierliche Bedingungen aufweisen, werden mit einer kombinierten Simulationform betrachtet. Diese Systeme treten wie bereits erwähnt am häufigsten auf. Drei fundamental unterschiedliche Interaktionstypen können zwischen diskreten und kontinuierlichen Parametern auftreten:

- Ein diskretes Ereignis kann eine diskrete Änderung eines kontinuierlichen Zustandsparameters hervorrufen.
- Ein diskretes Ereignis kann einen kontinuierlichen Zustandsparameter veranlassen sich zu einem bestimmten Zeitpunkt zu ändern.
- Wenn ein kontinuierlicher Zustandsparameter einen Grenzwert erreicht, kann ein diskretes Ereignis ausgelöst oder für einen späteren Zeitpunkt aktiviert werden.

Zeitfortschrittsmechanismen

Die dynamische Natur der Simulation mit diskreten Ereignissen erfordert das Mitführen des Zeitfortschritts in und während der Simulation. Dazu ist ein Mechanismus notwendig, der die Simulationszeit innerhalb des Modells sukzessive inkrementiert.

Im Generellen gibt es keinen Zusammenhang zwischen der simulierten Zeit innerhalb eines Modells und der Zeit, die ein Computer für die Simulationsdurchführung benötigt. Für die Simulation ist es folglich unerheblich wie schnell die simulierte Zeit, absolut betrachtet, abläuft. Erst wenn das Modell externe Objekte mit Echtzeit-Anforderungen integriert, wie zum Beispiel einen weiteren Computer oder Modelle mit Hardwareschnittstellen, muss die

simulierte Zeit äquivalent der realen Zeit ablaufen. Diese Echtzeit-Simulationen erfordern eine Zeitsynchronisation zwischen den Objekten des Modells und stellen insbesondere bei Simulationen mit Hardware-in-the-loop (Hardware im Simulationskreislauf) eine technische Herausforderung dar.

2.3 Modell-basierte Entwicklungs- und Verifikationsumgebung

In der Raumfahrt sind umfangreiche Systemtests zum Nachweis der vollen Funktionsfähigkeit des Raumfahrzeugs vor dem Raketenstart zu erbringen. Nach dem Start ist eine Reparatur in der Regel nicht mehr möglich, daher müssen alle möglichen Fehlfunktionen bereits im Vorfeld identifiziert und behoben werden. Dadurch wird explizit die Simulation der Weltraumbedingungen für die Sensorik des Raumfahrzeugs im Labor erforderlich.

Traditionell werden zu diesem Zweck Modelle des Raumfahrzeugs in Hardware erstellt, um Tests auf funktioneller, elektrischer und thermaler Ebene durchzuführen. Diese Modelle waren so ausgelegt, die elektrische oder thermale Systemspezifikation unter Weltraumbedingungen möglichst exakt abzubilden. Nur so konnten auf die jeweilige Thematik zugeschnittene Tests durchgeführt werden. Damit parallel an mehreren Tests gearbeitet werden konnte, wurden mehrere unterschiedliche Modelle erstellt.

Fertigung, Qualifikations- und Verifikationstests des Raumfahrzeugs stellen einen bedeutenden Faktor für die Entwicklungskosten und den Zeitplan dar [Larson & Wertz '99]. Weiterhin grenzen die verfügbare Testtechnologie und spezielle Anforderungen an die Einrichtungen die Machbarkeit des Projekts im Gesamten ein. In den aktuellsten Projekten mit hoher Komplexität ist der traditionelle Entwicklungsprozess kaum mehr finanzierbar und stößt auch an seine technischen Grenzen.

In jüngster Zeit gehen die Raumfahrtunternehmen daher immer mehr dazu über, die Methodik der computerunterstützten Systemsimulation in den Entwicklungs- und Verifikationsprozess zu integrieren. Die Senkung der Entwicklungskosten und die immer leistungsfähigeren Computer und Softwaresysteme sind die Gründe dafür. Neueste Entwicklungen in der Elektronik und Informationstechnologie eröffnen neue Perspektiven für den Entwicklungsprozess von Raumfahrzeugen. Der endgültige Funktionsnachweis wird nach wie vor am fertigen Flugmodell durchgeführt, aber die Systemsimulation kann die Entwicklungsmodelle bei gesteigerter Qualität und Aussagefähigkeit der Simulationstests nahezu vollständig ersetzen.

Seit 2001 entwickelt die EADS Astrium GmbH in Friedrichshafen am Bodensee eine modellbasierte Systemsimulationsinfrastruktur, um die Entwicklung von Raumfahrzeugen, die Verifikation der On-board Software und den Gesamtfunktionsnachweis eines Raumfahrzeugs zu unterstützen [Eickhoff et al. '03]. Diese Simulationsinfrastruktur wird „Model-based Development and Verification Environment” (MDVE, Modell-basierte Entwicklungs- und Verifikationsumgebung) genannt.

Parallel dazu ist ein neuer Entwicklungsprozess unter Nutzung der erwähnten Technologie entwickelt worden, bei dem auf Gesamtsystemebene komplett auf Entwicklungsmodelle verzichtet wird. In der jährlichen Vorlesungsreihe „Systemsimulation in der Satellitenentwicklung I & II“ von Dr. Jens Eickhoff [Eickhoff '08] wird dieser Prozess zusammenfassend beschrieben. In frühen Projektphasen 0/A und B (siehe Abbildung 2.4) wird jedes Untersystem separat entwickelt und es kommen dedizierte Simulatoren für die einzelnen Disziplinen und diverse Fragestellungen zum Einsatz. Schreitet das Projekt voran, tritt die funktionelle Interaktion zwischen allen Untersystemen im Hinblick auf zwei signifikante Aspekte in den Vordergrund: Der physikalischen und der kommunikativen Interaktion zwischen diesen Untersystemen. Beide Aspekte können nicht vollständig von den in den vorangegangenen Projektphasen verwendeten Simulationswerkzeugen getestet und verifiziert werden. Zum

Einen ist es jetzt möglich diese Aspekte mit kostspieligen Entwicklungsmodellen in Hardware zu untersuchen und verifizieren. Andererseits bietet die Systemsimulation durch den Austausch von einigen oder allen Hardwareelementen durch Softwaremodelle einen finanziellen und zeitlichen Vorteil: die Systemsimulation ermöglicht die Durchführung von Tests noch bevor die Flughardware des Raumfahrzeugs geliefert wird. Die Softwaremodelle werden dabei auf Basis der Systemspezifikationen des Herstellers und eigenen Untersuchungen erstellt.

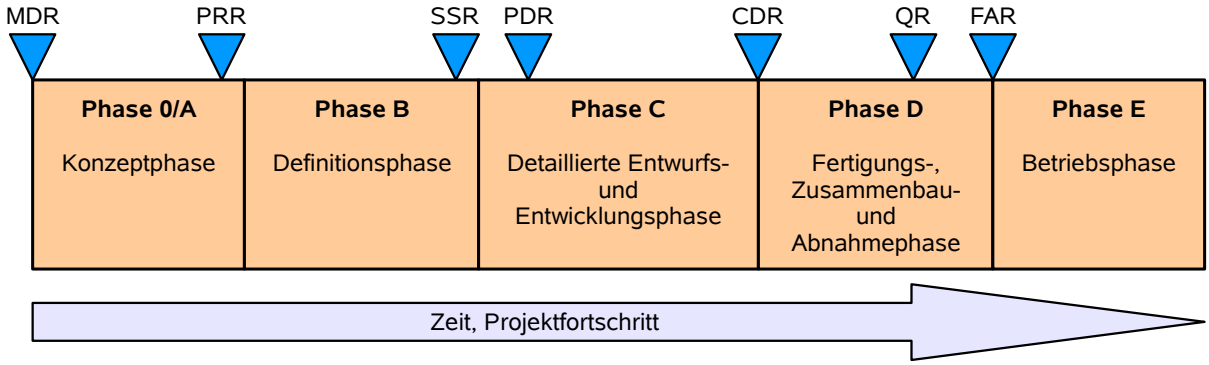


Abbildung 2.4: Unterteilung der Projektphasen in Raumfahrtprojekten nach Benz [Benz '07]

Die modell-basierte Entwicklung und Verifikation birgt eine außerordentliche Unterstützung zur Systemqualifikation und Funktionsverifikation. Die Systemsimulationsinfrastruktur erlaubt die Erstellung von Modellen des Raumfahrzeugs beginnend mit den ersten, rein auf Softwaresimulation beruhenden Testständen über hybride Teststände bis hin zu kompletten „FlatSat“ Testständen in den Projektphasen B, C/D und E eines Raumfahrtprojekts. Dabei werden die Teststände sukzessive durch Integration der On-board Software, der Flughardware und schließlich kompletter Flugprozeduren von reinen Softwaresimulationen zu „Hardware-in-the-loop“ Simulationen ausgebaut. Elemente aus vorangegangenen Konfigurationen können wiederverwendet bzw. angepasst und erweitert werden. Dadurch reduzieren sich gleichzeitig die Projektkosten und das Risiko immens. Die Vorteile dieses auf Systemsimulation basierenden Entwicklungsprozesses können wie folgt zusammengefasst werden:

- Die Modell-basierte Entwicklungs- und Verifikationsumgebung erlaubt dynamische Modellierung eines Raumfahrzeugs im Detail, mit dem Ziel der Analyse des Zusammenspiels der Komponenten auf funktionaler Ebene.
- Bezogen auf die Projektphase können bereits frühzeitig Simulationen durchgeführt werden, noch bevor die On-board Software oder Flughardware vorhanden ist. Dies gilt sofern die Systemkomponenten und das generelle Systemverhalten bereits definiert ist.
- Die On-board Software kann bereits getestet werden, bevor der On-board Computer in Hardware verfügbar ist.
- Flugprozeduren der On-board Software können weitgehend in softwarebasierten Testständen getestet und verifiziert werden. Auf späteren „Hardware-in-the-loop“ Testständen finden dann Timing Tests zwischen der Software und der kommunizierenden Hardware statt.
- Der Simulator kann als Teil des Missionskontrollsystems zu Trainingszwecken der Operatoren und als Softwarewartungseinrichtung genutzt werden.

Die MDVE Technologie ist bei EADS Astrium bereits in mehreren Projekten erfolgreich eingesetzt worden. Eine starke Tendenz, die rein auf Softwaremodellen basierenden Simulationen zu nutzen, ist durch die folgenden Vorteile gegeben:

- Die Entwicklung der numerischen Softwaremodelle ist schneller und preisgünstiger als die Erstellung von Hardware Teststänten.
- Die Softwaremodelle können einfacher an Designänderungen angepasst werden.
- Die rein aus Softwaremodellen basierten Teststände können ohne weiteres mehrfach installiert werden, um den Projektzeitplan zu beschleunigen. Durch die Tests werden keine Hardwarekomponenten blockiert.
- Es sind keine Transporte, Hardware-Zölle oder verwandte Logistikaufgaben notwendig.

2.3.1 Modell-basierte Systemsimulationsumgebung in der Satellitenentwicklung – Ein Überblick der wichtigsten Elemente und Funktionalitäten

Während der Entwicklung eines Raumfahrzeugs werden zu Testzwecken verschiedene MDVE Teststände in unterschiedlichen Konfigurationen verwendet. Diese Konfigurationen bauen aufeinander auf und sind jeweils für die konkrete Aufgabenstellung optimiert. Die hohe Wiederverwendbarkeit und Kompatibilität der einzelnen Komponenten ist durch die modulare Struktur der Software gegeben, wodurch sich die unterschiedlichen Konfigurationen zusammensetzen lassen.

Die verfügbaren Komponenten des MDVE Systemsimulators sind in Abbildung 2.5 illustriert und werden im Folgenden detaillierter erläutert.

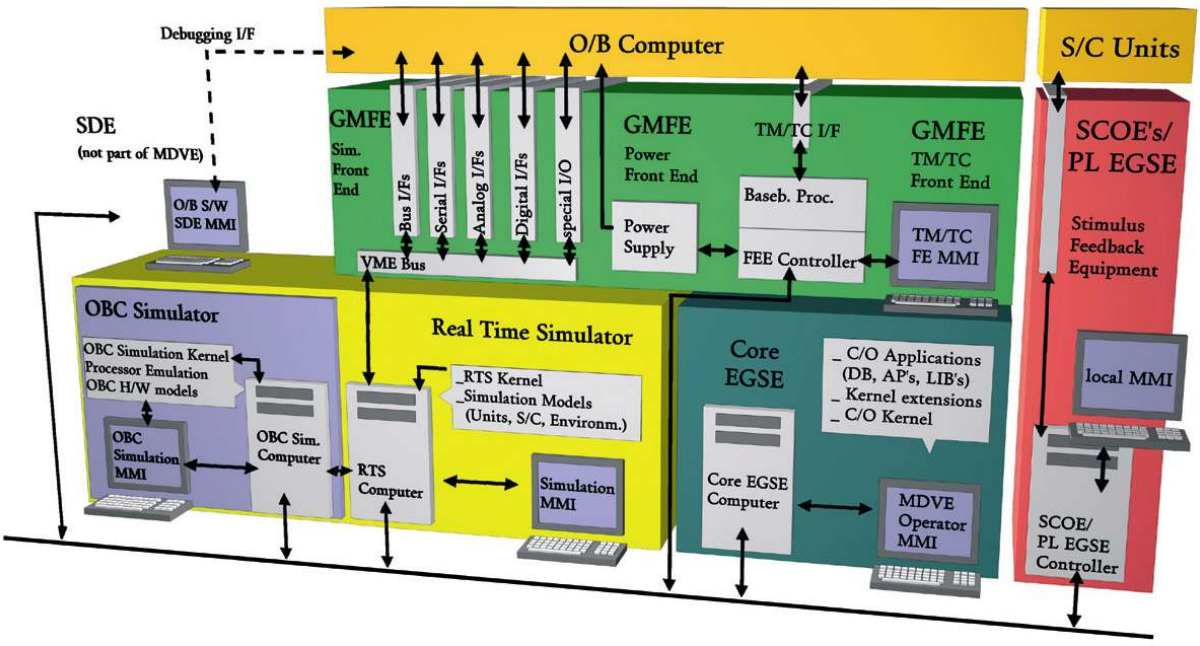


Abbildung 2.5: Komponenten des MDVE Systemsimulators [Eickhoff '08]

Das **Zentrale Kontrollsystem** (Core EGSE, Core Electrical Ground Support Equipment) ist eine Mensch-Maschine Schnittstelle (MMI, Man-Machine Interface), um alle relevanten Konfigurationen beginnend mit einer ersten reinen Software-Simulation über Hardware-in-the-loop Konfiguration bis hin zum fertigen Raumfahrzeug zu steuern. Über das Kontrollsystem wird sowohl der Simulator mit allen Komponenten als auch das virtuelle Raumfahrzeug bedient.

Das **modulare Frontend** (GMFE, Generic Modular Frontend) besteht aus Schnittstellenkarten, die den Datenverkehr zwischen den in Software simulierten Hardwaremodellen im Echtzeit-Simulator und den als Hardware-in-the-loop eingebundenen Geräten ermöglichen. Die Schnittstellenkarten werden von eigens erstellten speziellen Treibern gesteuert, die für diese Aufgabe optimiert wurden und ggf. Sonderfunktionen übernehmen.

Schließlich dient das **spezielle Testequipment** (SCOE, Special Check-out Equipment) zur Unterstützung von Hardware-in-the-loop Simulationen durch zusätzliche Kommunikationsgeräte. Über das spezielle Testequipment können die in Hardware in die Simulation eingebundene Satellitenkomponenten, wie zum Beispiel Sensoren mit ihren aktuellen Messdaten, stimuliert werden.

Der **On-board Computer Simulator** (OBC Simulator) ist ein eigenständiger Simulator der On-board Soft- und Hardware. Er kann sowohl durch ein einfaches funktionales Modell der On-board Software als auch durch eine komplette Implementierung der On-board Software auf einem Prozessoremulator repräsentiert sein und damit dem gesamten On-board Computer simulieren.

Kernelement aller MDVE Konfigurationen ist der **Echtzeit-Simulator** (RTS, Real Time Simulator). Seine Aufgabe ist die funktionale Simulation der Raumfahrzeughardware und der Weltraumumgebung. Dazu gehört das Organisieren der Datenströme, die Raumfahrzeugdynamik und Lagekontrolle und die Kontrolle des thermischen und elektrischen Energiehaushaltes. Die interne Struktur des Echtzeit-Simulator-Moduls ist in Abbildung 2.6 unten dargestellt und ist im Weiteren detaillierter erläutert.

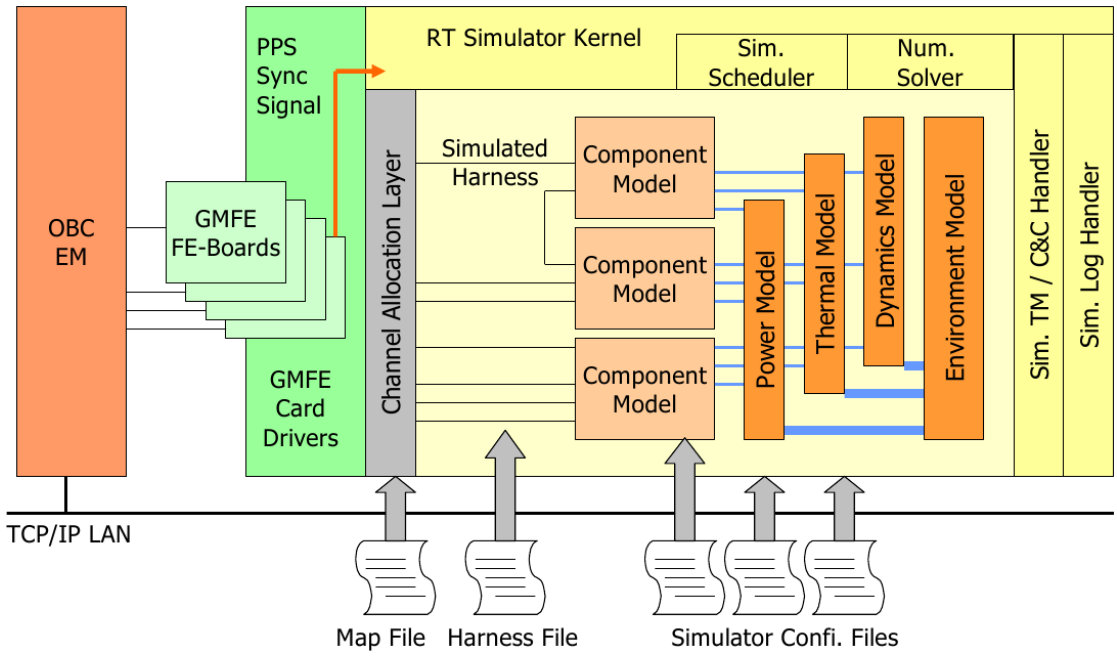


Abbildung 2.6: Interne Struktur des Echtzeit-Simulator-Moduls [Eickhoff '08]

2.3.1.1 Der Simulator-Kernel

Wichtigstes Element des Echtzeit-Simulators ist der Simulator-Kernel. Dieser ist sowohl modular als auch projektunabhängig aufgebaut und erlaubt das Anbinden von Hardware in den Simulationskreislauf. Der Kernel erfüllt unter anderem die im Folgenden beschriebenen Aufgaben.

Simulator-Steuerprogramm

Das Simulator-Steuerprogramm (Simulator Scheduler) steuert den Zeitfortschritt innerhalb des Echtzeit-Simulators und sorgt damit für einen Wechsel der Modellzustände aller implementierten Software Modelle. Die Modelle sind als C++ Klassen realisiert und besitzen eine standardisierte Struktur mit verschiedenen Basismethoden. Drei dieser Methoden werden vom Zeitfortschrittsmechanismus des Echtzeit-Simulators zyklisch aufgerufen. Beim Start des Simulators werden folgende Parameter über eine Konfigurationsdatei eingelesen und gesetzt:

- Die Basisperiode der Simulator-Aufrufliste. Sie stellt die Grundfrequenz dar, mit der die Liste aller implementierten und aufzurufenden Modelle abgearbeitet wird. Die Basisperiode muss kleiner oder gleich der kleinsten Modellaufrufperiode sein.
- Jedes an der Simulation teilnehmende Software-Modell ist durch einen Eintrag aufgeführt und wird somit berücksichtigt.
- Der Modelleintrag enthält die Modellaufrufperiode und einen optionalen Versatz. Die Modellaufrufperiode muss ein Vielfaches der Simulator-Basisperiode sein. Der optionale Versatz ermöglicht die Rechenlast des Computers über die Zeit zu verteilen.

Die Charakteristiken des Simulator-Steuerprogramms zeigen, dass der Echtzeit-Simulator für eine kombinierte diskret-kontinuierliche Simulation ausgelegt ist. Aus der Perspektive des Steuerprogramms treten Zustandsänderungen der Modelle zu diskreten und über der Zeit gleichmäßig verteilten Zeitpunkten auf. Aus der Perspektive des Modells selbst jedoch berücksichtigen mehrere Parameter, wie zum Beispiel die Drehzahl eines Reaktionsrades, das kontinuierliche Verhalten durch numerische Integration.

Der numerische Gleichungslöser

Der numerische Gleichungslöser repräsentiert den mathematischen Kernel. Er bietet zum Beispiel adäquate Werkzeuge für das Lösen von differentiellen Gleichungssystemen des Thermal- oder Dynamikmodells.

Simulator-Telemetrie und Kontrolle

Das Versenden von Simulator-Telemetrie und Verarbeiten von Kommandos wird vom Simulator-TM/TC Modul übernommen. Es interpretiert und übermittelt dem Simulator-Kernel Kommandos vom Zentralen Kontrollsystem und sendet selbigem neben der Telemetrie auch entsprechende Rückantworten.

Protokollierung

Die konfigurierbare Simulator-Protokollierung ermöglicht das zyklische Aufzeichnen von Parameterwerten.

2.3.1.2 Simulierter Kabelbaum

Die elektrischen Verbindungen zwischen den Hardware Bauteilen sind auch in der Echtzeit-Simulation durch Klassen und Methoden repräsentiert. Für jeden Leitungstyp gibt es eine eigene Klasse und entsprechende Methoden zur Funktionsabbildung dieser Leitung.

Simulierte Stromleitungen zum Beispiel besitzen unter anderem Parameter für die anliegende Spannung und den fließenden Strom. Weiterhin besitzt die Klasse Methoden, um diese Werte zu lesen und zu verändern.

Datenkommunikation kann durch die serielle Datenleitungsklasse realisiert werden. Da in der Regel bidirektionale Datenströme während eines Modellaufrufs vorhanden sind, implementiert die Klasse dies durch je eine Sende- und Empfangsleitung. Beide Leitungen arbeiten nach dem FIFO (First in, first out) Prinzip, d. h. das zuerst gesendete Datenpaket wird auch als erstes wieder ausgelesen.

Wichtig ist, dass von einem Modell gesendete bzw. auf Leitungen geschriebene Werte erst nach dem nächsten Zeitschritt anderen Modellen zur Verfügung gestellt werden. So ist der reale, sukzessive Datenfluss und nicht ein sozusagen gleichzeitiger Datenfluss berücksichtigt.

2.3.1.3 Physikalische Modelle der Simulator-Infrastruktur

Die physikalischen Modelle des Echtzeit-Simulators bilden keine Raumfahrzeughardware, sondern die Physik der Umgebung des Raumfahrzeugs ab. Die Softwaremodelle (Component Model) der einzelnen Raumfahrzeughardware interagieren mit diesen Umgebungsmodellen, wobei sie Umgebungsparameter wie Temperatur, Position und Bewegungszustand zur Verfügung stellen. Im Gegenzug übermitteln die Komponentenmodelle der Raumfahrzeughardware zum Beispiel die als Wärme dissipierte Energie oder aufgebrachte Momente der Aktuatoren an die Umgebungsmodelle. Zu den physikalischen Modellen gehören:

- Das Weltraumumgebungsmodell (Environment Model) berechnet die Parameter der das Raumfahrzeug umgebenden Weltraumbedingungen und stellt diese über ein Schnittstellenmodell allen Komponentenmodellen zur Verfügung. Das Umgebungsmodell wird beim Start des Simulators durch Konfigurationsdateien mit wichtigen Parametern und Konstanten (Erdgravitations- und Magnetfeld) versorgt, die zur Berechnung der vorherrschenden Weltraumbedingungen am gerade simulierten Ort zur simulierten Zeit benötigt werden.
- Das Dynamikmodell (Dynamics Model) bestimmt alle die Dynamik des Raumfahrzeugs beeinflussenden Parameter wie die aktuelle Position, die Lage und den aktuellen Bewegungszustand bzw. die Geschwindigkeit. Auch diese Parameter werden über ein Schnittstellenmodell allen Komponentenmodellen zur Verfügung gestellt. Die Differentialgleichungen des Bewegungszustandes werden durch den numerischen Gleichungslöser gelöst, wobei die Anfangswerte durch die Konfigurationsdateien vorgegeben werden.
- Die thermischen Zustände des Raumfahrzeugs werden im Thermalmodell bestimmt. Über eine Konfigurationsdatei ist ein vereinfachtes thermisches Modell des Raumfahrzeugs in Form eines Nichtlinearen Gleichungssystems definiert. Darin gegeben sind die Bilanzknoten mit ihrer Vernetzung untereinander und den zu berücksichtigenden Wärmeübertragungsmechanismen mit den jeweiligen Koeffizienten. Die Bilanzgleichungen beziehen sowohl die Wärmedissipation der Satellitenkomponenten

als auch die Wärmestrahlungsbedingungen des Weltraums mit ein. Nach dem Prinzip der Energieerhaltung können somit die Temperaturen an allen Bilanzknoten bestimmt werden.

- Im Modell der elektrischen Energie (Power Model) wird das Energiebudget verwaltet, d. h. die verbrauchte Energie gegenüber der in den Solarzellen generierten Energie aufsummiert und mit einer positiven oder negativen Batterieleistung gleichgesetzt. Anliegende Spannung und der fließende Strom in den einzelnen Leitungen sind hier die Eingabewerte der Energiebilanz.

Die Umgebungsmodelle sind Teil der Simulator-Infrastruktur und müssen wie die Komponentenmodelle im Simulator-Steuerprogramm zyklisch aufgerufen werden.

2.3.1.4 Die Komponentenmodelle

Jede Hardwarekomponente des Raumfahrzeugs ist im Simulator durch ein Komponentenmodell repräsentiert. Dabei liegt das Augenmerk auf einer funktionalen Abbildung aller für die Simulation relevanten Merkmale. Ziel des Modells ist es, die jeweiligen Eingaben des Modells (Kommandos, Umgebungsbedingungen oder Messsignale) in entsprechende Ausgaben (Status Informationen, Nutzdatentelemetrie, induzierte Kräfte und Momente) zu verarbeiten und weiterzugeben. Dazu besitzen die Modelle Zustandsparameter und mathematische Algorithmen bzw. Bedingungen, um zusammen mit den Modelleingaben zu einer definierten Zustandsänderung zu gelangen. Es hat sich bewährt das Modell in einen diskreten und einen kontinuierlichen Bereich zu unterteilen.

- Der diskrete Teil ist ein Zustandsautomat. Er ist verantwortlich für die Verarbeitung der Modelleingaben, die entsprechende Änderungen der Zustandsparameter, den Umgang mit induzierten Fehlerfällen und für die Bereitstellung der Modellausgaben. In Abhängigkeit der Komplexität kann eine Komponente in mehreren Untermodellen realisiert werden, deren diskrete Modellanteile sehr klar und eindeutig abgebildet sind.
- Der kontinuierliche Teil birgt das mathematische Modell einer veränderlichen physikalischen Größe (Kräfte, Momente) und wird innerhalb des diskreten Modellanteils ausgeführt. Diese nicht-diskrete physikalische Größe wird typischerweise durch Integration über die Modellaufrufperiode bestimmt und entweder exakt oder mit einem Durchschnittswert weiterverarbeitet.

Die Komponentenmodelle sind jeweils eigene C++ Klassen, die bei der Simulatorinitialisierung durch Konfigurationsdateien mit entsprechenden Parameterwerten versorgt werden. Durch dieses Verfahren lassen sich Instanzierungen realisieren, d. h. ein und dieselbe Klasse wird mehrmals mit unterschiedlichen Parameterwerten erzeugt. Dies ist zum Beispiel bei identischen, aber im Raumfahrzeug unterschiedlich montierten und angeschlossenen Reaktionsrädern oder Sensoren der Fall.

Verifiziert wird die Funktionalität der Komponentenmodelle in sogenannten Baugruppentests, d. h. das Komponentenmodell wird individuell und losgelöst von den anderen Modellen getestet. Dieses Vorgehen ist in der Computertechnologie verbreitet, da so, angefangen bei den kleinsten testfähigen Modulen bis hin zum gesamten System, ein eigenständiger Funktionsnachweis erbracht werden kann. Im Kontext des MDVE Systemsimulators ist ein einzelnes Komponentenmodell die kleinste testfähige Baugruppe. Die Tests schließen sowohl den diskreten als auch den kontinuierlichen Modellanteil ein.

2.3.1.5 Randbedingungen des Modellaufrufs

Modellaufrufperiode und Versatz definieren den Modellaufruf der Komponentenmodelle und beeinflussen das Verhalten des gesamten simulierten Systems. Jede Komponente ist funktionaler Teil eines offenen oder geschlossenen Kreislaufs. Das Lageregelungssystem zum Beispiel besteht aus Sensoren, Regelgliedern und Aktuatoren, die einen geschlossenen Kreislauf bilden. Auf der anderen Seite ist die Sammlung und Weiterleitung von wissenschaftlichen Daten mit Kameras beispielsweise ein offener Kreislauf.

Das Abbilden von Hardwarekomponenten bedeutet damit auch das Modellieren von deren Regelkreisläufen. Bei der Modellausführung werden zum Einen die funktionellen Modellmethoden abgearbeitet und zum Anderen mit den übrigen Komponentenmodellen über die Leitungen kommuniziert. Über das Simulator-Steuerprogramm werden die Modelle mit einer definierten Aufrufperiode ausgeführt. Durch die geeignete Wahl von Aufrufperiode und Versatz muss sichergestellt werden, dass der Regelkreislauf von Sensoren über Regelglieder zu den Aktuatoren und wieder zurück zu den Sensoren innerhalb eines Regelintervalls ausgeführt wird, d. h. die jeweiligen Modelle nacheinander ausgeführt werden und jeweils dazwischen ein Datenaustausch erfolgt ist. Abbildung 2.7 veranschaulicht diesen Zusammenhang am Beispiel des Lageregelungssystems mit den Systemkomponenten des *Flying Laptop*.

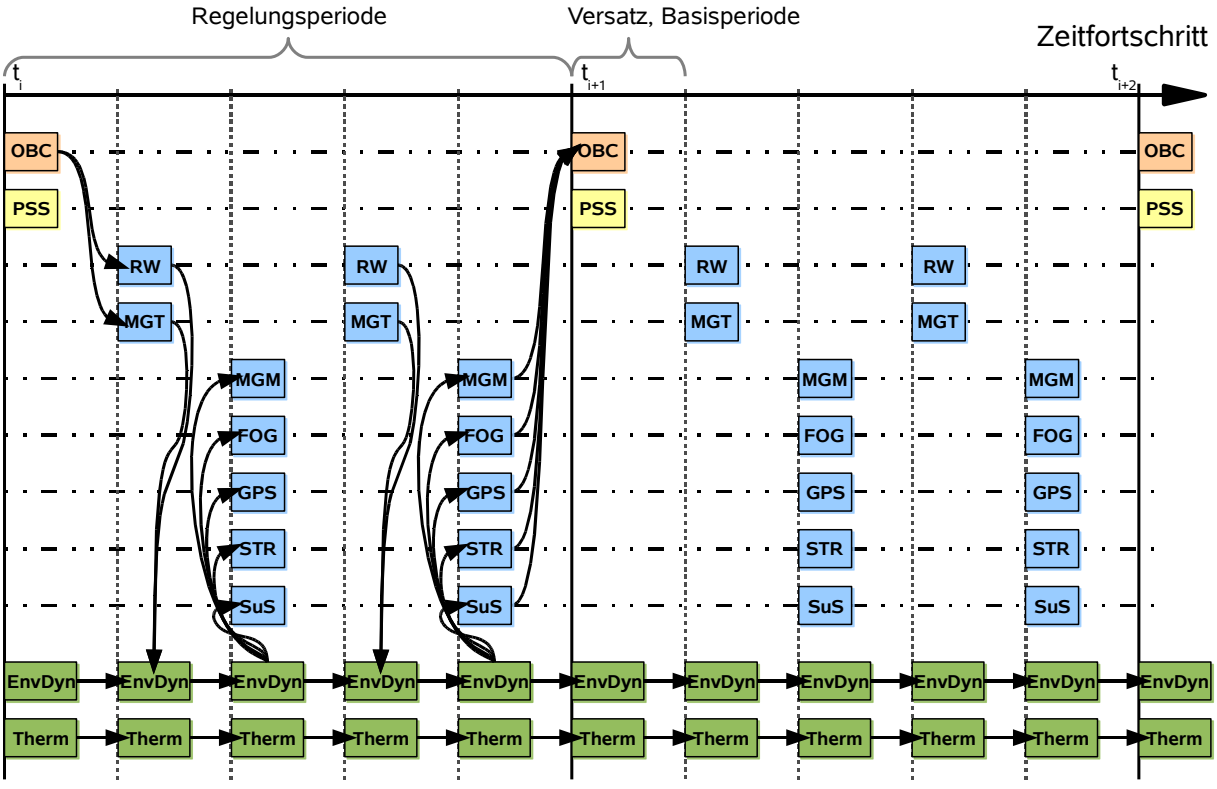


Abbildung 2.7: Ablaufreihenfolge und Datenfluss zwischen Modellen des Lageregelungssystems; hier am Beispiel der Systemkomponenten des Kleinsatellitenprojekts Flying Laptop

Der Datenfluss zwischen den Komponentenmodellen erfolgt jeweils mit dem Zeitfortschritt. Bei der Ausführung der Lageregelungsalgorithmen in der On-board Computer Software werden Kommandos an die Stellglieder Reaktionsrad (RW) und Magnetorquer (MGT) erzeugt und übertragen. Diese Kommandos werden beim nächsten Aufruf der Stellglieder, basierend auf dem bisherigen Zustand in aktualisierte Stellgrößen wie die Drehzahlbeschleu-

nigung der Reaktionsräder und des korrespondierenden Stellmoments, umgerechnet und als Zustandsgrößen an das Umgebungs- und Dynamikmodell (EnvDyn) gegeben. Dieses wiederum berechnet mit dem Impulssatz basierend aus den bisherigen Ist-Größen des Bewegungszustands und aus allen angreifenden Kräften und Momenten die neue Position, Lage und den Bewegungszustand des Raumfahrzeugs. Im nächsten Zeitschritt werden die aktuellen Größen der Position, Lage und des Bewegungszustands von den Sensoren Magnetometer (MGM), den Faser-optischen Kreiseln (FOG), dem Globalen Positionsbestimmungssystem (GPS), den Sternensensoren (STR) und den Sonnensensoren (SuS) abgefragt. In den Sensormodellen wird aus den erwähnten Größen das komponentenspezifische Datenpaket erzeugt und auf Anfrage oder zyklisch an den On-board Computer gesendet, der die Informationen wieder für die Verarbeitung der Lageregelungsalgorithmen benötigt. Dadurch wird der Simulationskreislauf geschlossen und beginnt wieder von vorn.

Nach dem Abtasttheorem von Shannon [Wikipedia '08b] müssen jedem Aufruf der Regelglieder mindestens zwei Aufrufe der Sensoren bzw. Aktuatoren vorangegangen sein. Dies führt in diesem Beispiel dazu, dass sich nach dem ersten kaskadiertem Aufruf von Aktuatoren und Sensoren eine zweite Kaskade anschließt und erst in dieser zweiten Kaskade neue Informationen für die Lageregelungsalgorithmen in der On-board Software bereitgestellt werden. Für eine weitere Erhöhung der Simulationsstabilität kann die Anzahl der Kaskaden erhöht werden.

2.3.1.6 Simulator Konfiguration

Beim Start des Simulators findet eine Initialisierung aller Parameter statt, deren Werte aus einer Reihe von Konfigurationsdateien eingelesen werden. Die Konfigurationsdateien sind als XML-Dateien realisiert. Über diesen Mechanismus ist sowohl der Simulator selbst als auch explizite Simulationsszenarien durch ihre Startwerte konfigurierbar. Für jedes zu simulierende Missionsszenario müssen die Dateien konsistent erstellt werden.

Die Konfigurationsdateien kann man in Standarddateien, die immer identisch sind, und in je nach Missionsszenario wechselnden charakterisierende Dateien aufteilen. Letztere werden zum Schluss eingelesen und können jeden bereits aus der Standardkonfiguration gesetzten Parameterwert nochmals mit einem missionsszenario-spezifischen Wert überschreiben. Somit ist es möglich einen bestimmten Startzustand des Missionsszenario zu erreichen bzw. bewusst ein Fehlverhalten einzelner Systeme zu induzieren und anschließend die Reaktion des Gesamtsystems zu beobachten.

2.3.2 Teststand Konfigurationen der Systemsimulationsumgebung

Die Struktur der MDVE Systemsimulationsumgebung ist modul-basiert, d. h. die Hard- und Softwarekonfiguration des Teststandes kann den Testzielen entsprechend beliebig zusammengesetzt werden. Die Summe der möglichen Teststände kann in die folgenden beiden Kategorien aufgeteilt werden:

- Software-Verifikations-Einrichtung (SVF, Software Verification Facility)
- Echtzeit-Teststand (RTB, Real Time Testbed)

Nachfolgend werden die beiden für das Verständnis dieser Dissertation wichtigsten Teststände detaillierter erläutert.

2.3.2.1 Software-Verifikations-Einrichtung

Die Software-Verifikations-Einrichtung (SVF) simuliert die komplette Satellitenhardware in Softwaremodellen auf einem oder mehreren Computern. Hierbei wird jede Hardware Einheit durch ein Softwaremodell repräsentiert, welches das exakte Verhalten der Hardware abbildet. Es handelt sich folglich um eine reine Simulation in Software, die durch Modelle der Weltraumumgebung und Dynamik, des Thermalsystems und des Energieversorgungssystems unterstützt wird.

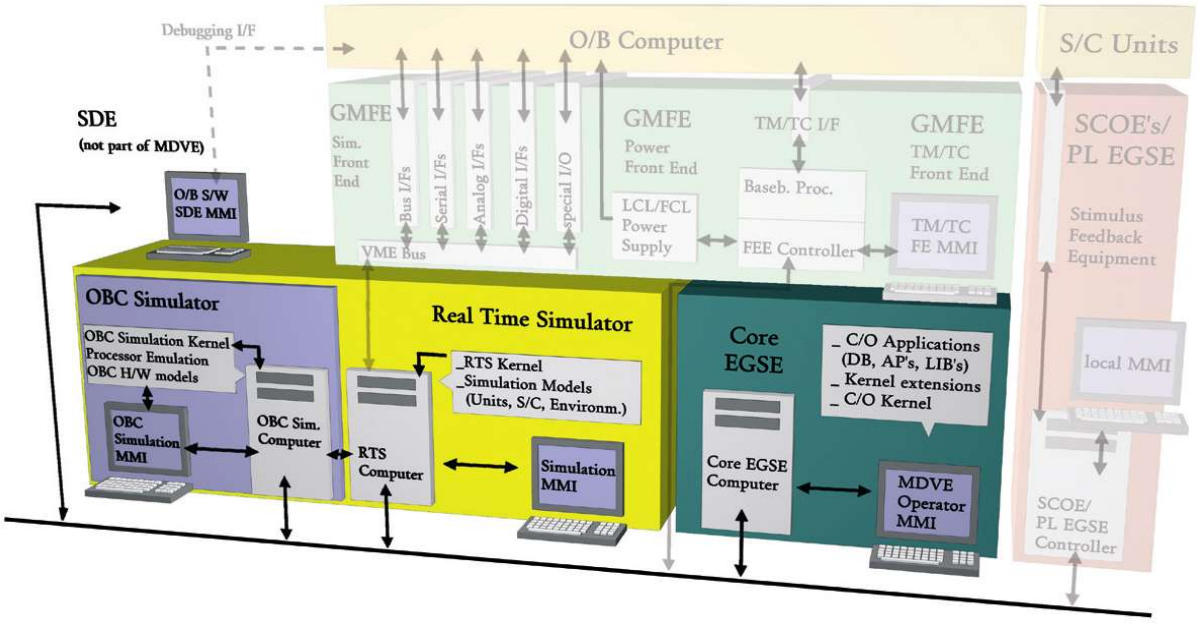


Abbildung 2.8: MDVE Konfiguration: Software-Verifikations-Einrichtung [Eickhoff '08]

Wie in Abbildung 2.8 ersichtlich, besteht der Simulator in dieser Konfiguration aus dem Echtzeit-Simulator (Real Time Simulator) und dem On-board Computer Simulator (OBC Simulator). Das Zentrale Kontrollsystem (Core EGSE) bietet darüber hinaus eine Telemetrie- und Telekommando-Schnittstelle, sowie die Simulator-Steuerung und Protokollierung.

Die funktionelle Verifikation des Datenflusses, Kommando-Sequenzen und On-board Computer Software Fehlerbeseitigung und Vor-Verifikation sind nur ein Auszug der Aufgaben, die mit einer SVF erfüllt werden können. Weiterhin können Simulationen kompletter Missionsszenarien durchgeführt werden.

Alle Simulation werden unter Echtzeit-Bedingungen ausgeführt. Dies bedeutet, dass jede Aufgabe in einem vordefinierten Zeitintervall erledigt wird. Es bedeutet jedoch nicht, dass die simulierte Zeit nicht schneller als die reale Zeit laufen kann. Da der Zeitfortschritt für alle beteiligten Softwaremodelle simuliert wird, können so gegenüber der realen Zeit auch beschleunigte oder verlangsamte Simulationen stattfinden.

2.3.2.2 Echtzeit-Teststand

Sobald in einem Echtzeit-Teststand (RTB) Hardware in die Simulation integriert wird, muss diese auch gegenüber der realen Zeit unter Echtzeit-Bedingungen durchgeführt werden, da das Zeitverhalten der Hardware in der Regel nicht beeinflusst werden kann. Echtzeit-Teststände treten in verschiedenen Konfigurationen auf:

- Echtzeit-Teststand: Der On-board Computer ist neben dem Test-Kabelbaum das einzige Hardwareelement in der Simulation. Die restlichen Satellitenkomponenten sind im Echtzeit-Simulator in Software abgebildet. Dieser Teststand dient der Verifikation des On-board Computers und der On-board Software.
- „FlatSat“-Teststand: Ein oder mehrere Satellitenkomponenten oder ihre Entwicklungsmodelle sind in die Simulation integriert. Dies ist entweder durch einen geschlossenen oder offenen Simulationskreislauf möglich. Der Name „FlatSat“ deutet auf die flache Anordnung der Komponenten, die einfach auf einem Tisch ausgebreitet und durch einen Test-Kabelbaum angeschlossen sind. Primäres Ziel der Tests ist die Hardware ↔ Hardware-Kompatibilität der beteiligten Komponenten.
- Protoflight-Teststand: Die Satellitenkomponenten sind bereits in der Satellitenstruktur montiert. Ein Test-Kabelbaum verbindet nahezu alle Komponenten mit dem Simulator, um so weitere Tests der Hardware durchführen zu können.

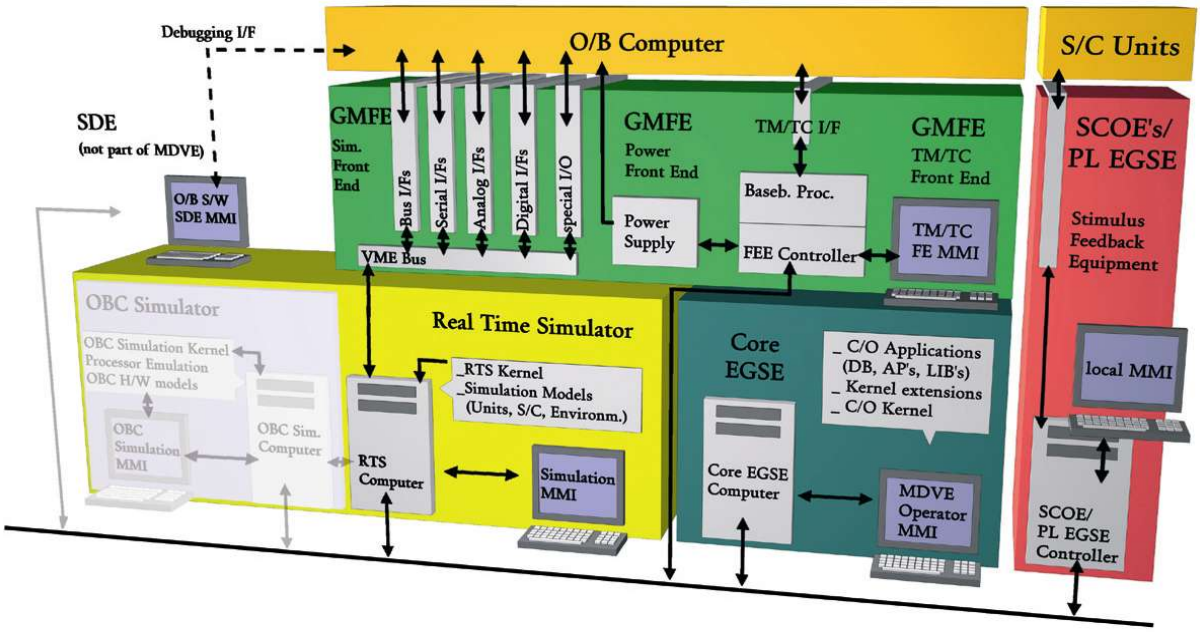


Abbildung 2.9: MDVE Konfiguration: Echtzeit-Teststand [Eickhoff '08]

In allen Konfigurationen ist der On-board Computer oder sein Entwicklungsmodell mit dem Echtzeit-Simulator durch den Test-Kabelbaum und sogenannten generisch-modularen Front-end Schnittstellenkarten (GMFE) verbunden (siehe Abbildung 2.9). In der Simulation ersetzt er jetzt den On-board Computer Simulator. Die GMFE Karten stellen die entsprechende elektrische Kommunikationsschnittstelle zwischen dem On-board Computer und dem Simulator bereit. Spezielles Testequipment (SCOE) wird verwendet, um Hardware-spezifische Aufgaben, wie zum Beispiel die Energiegeneration der Solarzellen, zu erfüllen oder aber, um die Satellitensensoren entsprechend der simulierten Weltraumbedingungen zu stimulieren.

2.4 Softwaretechnische Voraussetzungen

Die Modell-basierte Entwicklungs- und Verifikations-Technologie benötigt eine Vielzahl an Softwarewerkzeugen. Nicht alle davon sind als unbedingte Voraussetzungen zu verstehen, dennoch steigern sie die Produktivität und führen letztlich zu einer höheren Produktqualität. Die folgenden Absätze geben einen Überblick und Referenzen über die wichtigsten Werkzeuge und ihre Aufgaben im Kontext modellbasierter Entwicklung im Team.

2.4.1 Concurrent Versions System

Das Concurrent Versions System (CVS) [CVS '86] ist ein quelloffenes Versionskontrollsystem. Mit ihm können alle Erweiterungen und Änderungen in einer Gruppe von Dateien, typischerweise eines Softwareprojekts nachverfolgt und verwaltet werden. Es erlaubt vielen Entwicklern unabhängig von der Nähe ihrer Arbeitsplätze zueinander zusammenzuarbeiten. Ausgedacht und entwickelt wurde es in den 1980ern von Dick Grune, der gemeinsam mit einigen seiner Studenten an einem Softwareprojekt arbeitete. In der Open Source Software Gemeinde ist CVS sehr beliebt geworden. Es wird unter den Bedingungen der GNU General Public License [GPL '91] herausgegeben, die unter der Betreuung der Free Software Foundation (FSF) [FSF '85] steht.

CVS ist ein produktionsqualität steigerndes System mit nicht reserviertem Quelltext, d. h. es erlaubt mehr als einem Entwickler gleichzeitig an denselben Dateien zu arbeiten. Dem Prinzip der parallelen und gemeinsamen Entwicklung von Software wird es nicht nur durch die Struktur aus einem Server-Datendepot und einer beliebigen Anzahl von Nutzerterminals gerecht, sondern auch durch die Fähigkeit, einzelne unabhängige Änderungen an ein und derselben Datei zu einer finalen Version zusammenzuführen. Einige andere kleine freie Werkzeuge unterstützen CVS bei seinen Aufgaben, zum Beispiel dem Identifizieren von Unterschieden zwischen zwei Dateien oder aber der Darstellung dieser Differenzen direkt nebeneinander.

Dank standardisierter Kommunikationsprotokolle können CVS Server und Nutzerterminals mit unterschiedlichen Operationssystemen, wie zum Beispiel Microsoft Windows oder einem Unix Derivat, arbeiten. Das Kommandozeilenprogramm bzw. graphische Oberflächen sind für beide Betriebssysteme auf Basis der GPL verfügbar.

2.4.2 Datenbanksysteme

„Ein Datenbanksystem ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe eines Datenbanksystems ist es, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen.

Ein Datenbanksystem besteht aus zwei Teilen: der Verwaltungssoftware, genannt Datenbankmanagementsystem (DBMS) und der Menge der zu verwaltenden Daten, der eigentlichen Datenbank. Die Verwaltungssoftware organisiert intern die strukturierte Speicherung der Daten gemäß einem vorgegebenen Datenbankmodell (z. B. dem relationalen Datenbankmodell) und kontrolliert alle lesenden und schreibenden Zugriffe auf die Datenbank. Als

externe Schnittstelle stellt sie eine Datenbanksprache zur Formulierung von Abfragen, zum Einfügen und Ändern von Daten und für administrative Befehle zur Verfügung. Die Datenbank enthält zusätzlich zu den eigentlichen Daten noch die Beschreibung der Daten, den so genannten Datenkatalog.“ [Wikipedia '08a]

Im Kontext dieser Dissertation wurde ein Datenbanksystem basierend auf dem quelloffenen Datenbankmanagementsystem MySQL [MySQL '96] installiert und genutzt. Es ist ebenfalls unter den Lizenzbedingen der GNU GPL frei verfügbar.

Weitere unter Unix/Linux bzw. Microsoft Windows angewandte meist kommerzielle Datenbanksysteme sind in Dicken et al. *„Datenbanken unter Linux“* [Dicken et al. '00] zu finden und zu detaillierteren Erklärungen sei darauf verwiesen.

2.4.3 Extensible Markup Language

Die Extensible Markup Language (XML) ist eine erweiterbare Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Textdateien. Die XML Spezifikation [XML '98] ist vom World Wide Web Consortium (W3C) [W3C '94] in ihrer ersten Fassung bereits 1998 herausgegeben worden und liegt momentan in der vierten Ausgabe vom 29. September 2006 vor. Sie definiert eine Metasprache, die durch strukturelle und inhaltliche Einschränkungen eine anwendungsspezifische Sprache bzw. eine Datenstruktur definiert.

Im Rahmen der XML Spezifikation kann die Form der Datenstruktur selbst definiert und diese dann als Grundlage der Datenmenge verwendet werden. Durch diese hohe Flexibilität findet XML breite Anwendung in Computer Systemen. Als gemeinsame Schnittstelle wird XML in der Regel für den Austausch von Daten zwischen unterschiedlichen Anwendungen und Systemen verwendet.

2.4.4 Unified Modeling Language

Die Unified Modeling Language (UML) [UML '97] ist eine von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software und anderen Systemen. Als allgemein verwendbare Modellsprache besitzt die UML Definition graphische Notationen, um abstrakte Modelle von kompletten oder Teilsystemen zu erstellen, die dann als UML-Modelle gelten. Mit Hilfe von UML werden Software- oder andere Systeme spezifiziert, visualisiert, entwickelt oder dokumentiert.

Während der letzten Jahre hat die UML in ihrer zweiten Version einen großen Teil zur Entwicklung der modellgestützten Technologien beigetragen. Die Einführung von industriell einvernehmlichen Definitionen der graphischen Notation zur Darstellung von allgemeinen Objekten (wie zum Beispiel Klassen, Komponenten, Generalisation, Aggregation) und Verhaltensweisen erlaubte den Software Entwicklern sich stärker auf Design und Architektur konzentrieren zu können. Gleichermaßen gilt dies für ingenieurtechnische Modelle, die Darstellung von Geschäftsprozessen oder organisatorischen Strukturen.

Zur detaillierteren Erklärung des UML Standards, der Regeln und Notationen sei auf Neumann *„Objektorientierte Softwareentwicklung mit der Unified Modelling Language (UML)“* [Neumann '02] und Fowler *„UML konzentriert“* [Fowler '03] verwiesen.

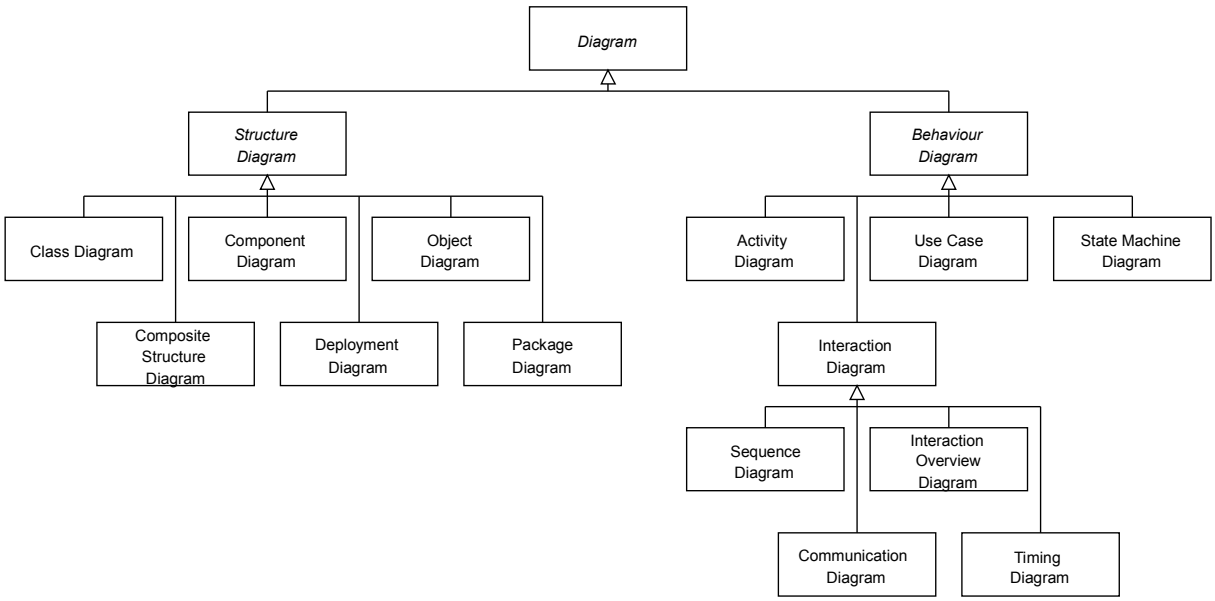


Abbildung 2.10: Die UML 2.0 Diagrammtypen in hierarchischer Darstellung [Ryan '07]

UML in der Version 2.0 besitzt dreizehn unterschiedliche Diagrammtypen zur Verdeutlichung von Zuständen, Beziehungen und Abläufen. Abbildung 2.10 zeigt diese in hierarchischer Darstellung. Es gibt zwei Hauptkategorien von Diagrammtypen: Struktur-Diagramme und Verhaltensdiagramme, deren wichtigste Unterkategorie die Interaktionsdiagramme repräsentieren. Struktur-Diagramme zeigen alle Objekte des modellierten Systems, während Verhaltensdiagramme notwendige Vorgänge des Modells darstellen. Schließlich verdeutlichen die Interaktionsdiagramme die Ablaufkontrolle und den Datenfluss zwischen den Objekten im Modell.

2.4.5 Wiki

Ein Wiki ist eine gemeinschaftliche Webseite, die von jedem Zugriffsberechtigten direkt verändert werden kann. Die Wikiseiten sind sowohl beim Lesen als auch beim Bearbeiten von beliebiger Information einfach und intuitiv gestaltet. Eines der meist bekannten Wiki's ist Wikipedia [Wikipedia '01].

Das Institut für Raumfahrtsysteme hat initiiert durch die Systemsimulations-Gruppe im Jahr 2005 ein eigenes Wiki namens `MoonWiki` [MoonWiki '06] gestartet. Das Erscheinungsbild der Startseite des Stuttgarter `MoonWiki` ist in Abbildung 2.11 unten illustriert. Installiert mit dem quelloffenen Softwarepaket MediaWiki [MediaWiki '03], unterstützt es allgemeine Institutsaufgaben und im speziellen das Stuttgarter Kleinsatelliten Programm. Im World Wide Web (WWW) ist es unter der folgenden URL verfügbar:

```
http://moonwiki.irs.uni-stuttgart.de/
```

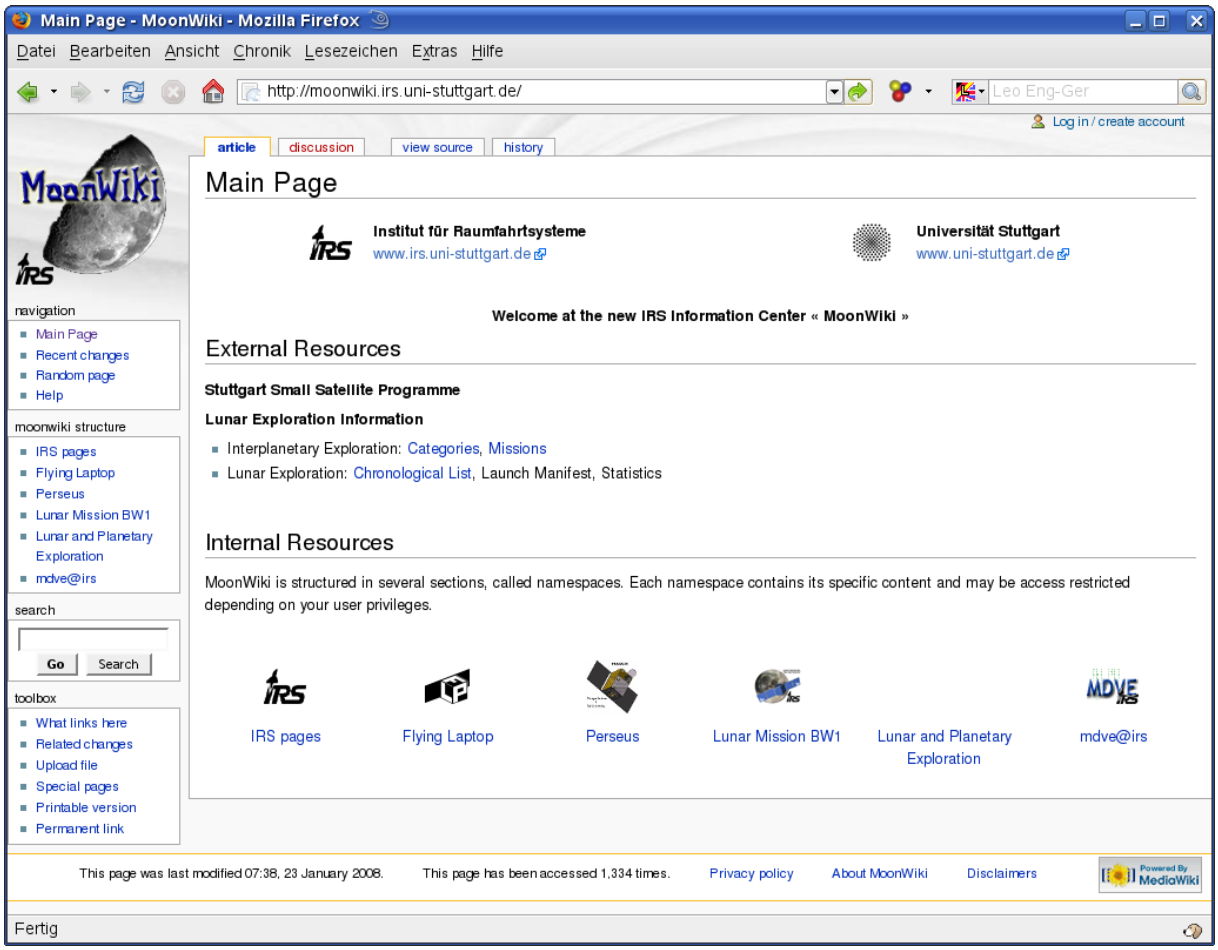


Abbildung 2.11: Startseite des MoonWiki unter http://moonwiki.irs.uni-stuttgart.de/

Sukzessiv erweiterte Informationen wie zum Beispiel über wichtige Konfigurationsoptionen des Nutzerkontos, Tipps und Umgang mit verwendeter Software oder organisatorische Abläufe usw. sind im Wiki in mehrere projektspezifische Sektionen, so genannte Namespaces, aufgeteilt.

Der bedeutendste Vorteil eines Wiki Systems gegenüber eher statischen Geschäftsdokumenten ist die einfache, schnelle, unformale und dauerhafte Art des Informationsaustausches. Jeder registrierte Nutzer kann Informationen hinzufügen, korrigieren, verbessern oder die Wiki Wissensdatenbank erweitern und dieses Wissen ist anschließend jedem Nutzer durch die bekannten Suchfunktionen einfach zugänglich. Dem Problem von flüchtigem Wissen und Erfahrungen durch die starke Fluktuation der beteiligten Studenten in Universitätsprojekten kann damit einfach entgegengewirkt werden.

Während der Einarbeitung in die von EADS Astrium GmbH zur Verfügung gestellten Simulatorelemente hat speziell die Systemsimulations-Gruppe durch das Wiki Prinzip des Informationsaustausches großen Vorteil daraus gezogen. Auch bei der Erstellung von eigenen Modellen und schließlich dem Umgang mit dem vollständigen Simulator war das Wiki eine große Hilfe für das Team. Das Ziel war und ist ein stetiges Anwachsen dieser Wissensdatenbank parallel zum Entwicklungsprozess des Kleinsatelliten Programms.

Sicherheit und Zugriffsschutz sind in diesem Zusammenhang ein wichtiges Thema, welches auch aufgrund der sensiblen Informationen nicht vernachlässigt werden darf. Zu diesem Zweck ist eine Erweiterung des Wiki-Systems installiert, die erweiterten Zugriffsschutz und

dessen Verwaltung ermöglicht. Dadurch benötigt jeder Nutzer die Zugriffserlaubnis auf den Namespace der aufgerufenen Seite. Ist die Zugriffserlaubnis nicht gegeben, wird die Seitendarstellung blockiert. Die Zugriffserlaubnis muss jedem Nutzer explizit von einem Systemadministrator, der ein Angestellter des Instituts für Raumfahrtsysteme ist, erteilt werden. Weiterhin können neue Accounts nur durch den Systemadministrator angelegt werden. Ist der Nutzer weder angemeldet noch registriert, so kann er nur auf ein paar wenige allgemeine Seiten zugreifen und im Speziellen liefern Suchanfragen nur Ergebnisse von den Seiten, auf die der Nutzer Zugriff hat.

3 Durchführung und Umsetzung

Einen wesentlichen Anteil an der Umsetzung der in dieser Dissertationen behandelten Aufgabenstellung stellt der Aufbau und die Inbetriebnahme des Systemsimulators als Werkzeug zur Systemsimulation im Allgemeinen dar. Im ersten Unterkapitel wird kurz auf die damit verbundene Hardware und Software-Infrastruktur eingegangen. Erst im nächsten Schritt ist das Werkzeug für den Anwendungsfall innerhalb des Kleinsatellitenprojekts *Flying Laptop* vorbereitet und die konkrete Entwicklung von Teststänten beginnt. Das zweite und dritte Unterkapitel behandeln die realisierten Teststände und beispielhaft ein paar der dazu benötigten und entwickelten Satellitenkomponentenmodelle. Der letzte Abschnitt befasst sich mit der Inbetriebnahme des Missionskontrollsystems zur Steuerung und Überwachung der Simulation.

3.1 Hard- und Software Entwicklungs-Infrastruktur

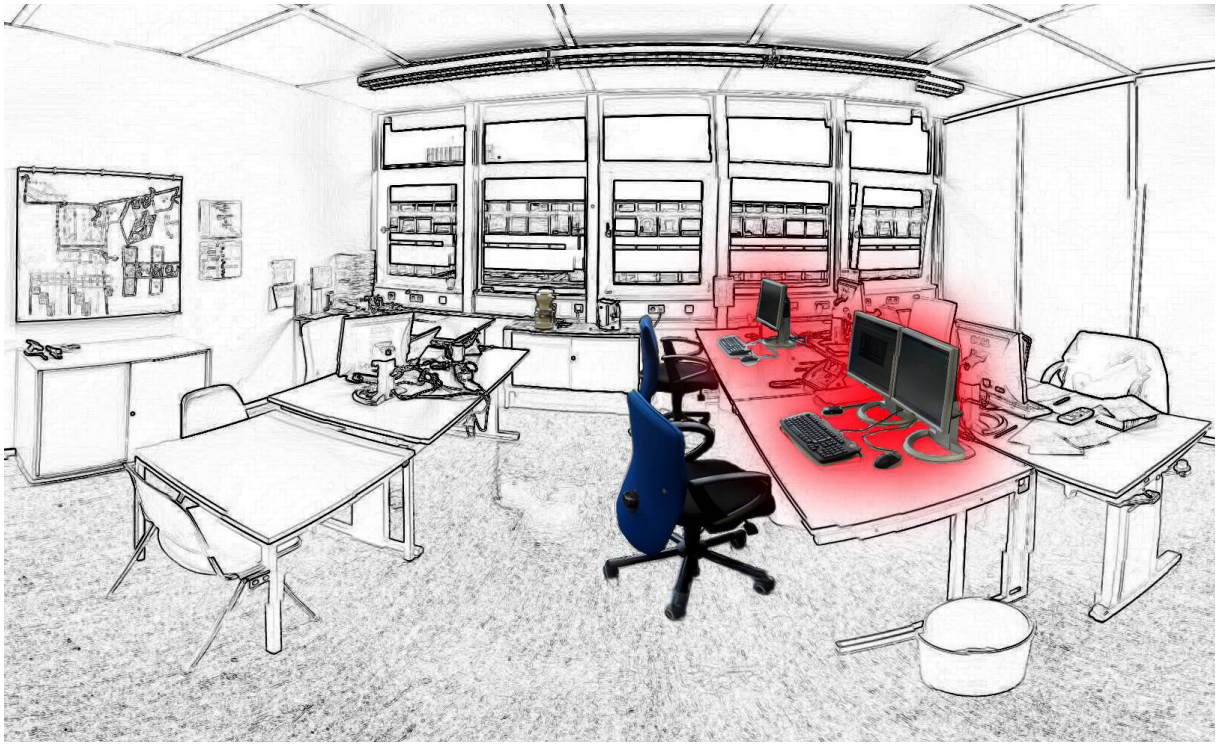


Abbildung 3.1: Das MDVE-Labor am IRS mit farblich hervorgehobenem Teststand

Mit dem Begriff System-Simulationsumgebung ist unter anderem eine umfangreiche Simulationssoftware gemeint, die auf einer kleinen Anzahl von Personal Computer (PC) entwickelt, gewartet und gesteuert wird. In diesem Unterkapitel wird näher auf die Umsetzung der Hardware und Software-Infrastruktur der MDVE-Installation am Institut für Raumfahrtsysteme eingegangen. Räumlich ist die MDVE Thematik im sogenannten MDVE-Labor gebündelt, welches am IRS alle Entwicklungsarbeitsplätze bereitstellt. Für den Aufbau von „FlatSat“ bzw. Echtzeit-Teststänten ist ein Bereich im Satelliten-Integrationslabor (SIL) frei gehalten.

3.1.1 Computer Hardware

In Abbildung 3.2 unten ist die gesamte benötigte Computer Hardware Infrastruktur graphisch veranschaulicht. Rückgrat der Computerausstattung ist ein Server, der als Datenbank-, Datei- und Software License-Server fungiert. Die Projektdatenbank, das UML Repositorium und der gesamte MDVE Quelltext in C++ ist auf dem Server zentral gespeichert und steht allen anderen Computern im MDVE-Labor zur Verfügung. Die sichere Datenspeicherung ist durch ein RAID Level 5 Festplattensystem mit fünf aktiven und einer Ersatzfestplatte garantiert. Zur zuverlässigen Datensicherung und Wiederherstellung wird zusätzlich täglich ein verschlüsseltes Backup auf ein physikalisch und räumlich unabhängiges Sicherungssystem im Universitätsnetzwerk erstellt.

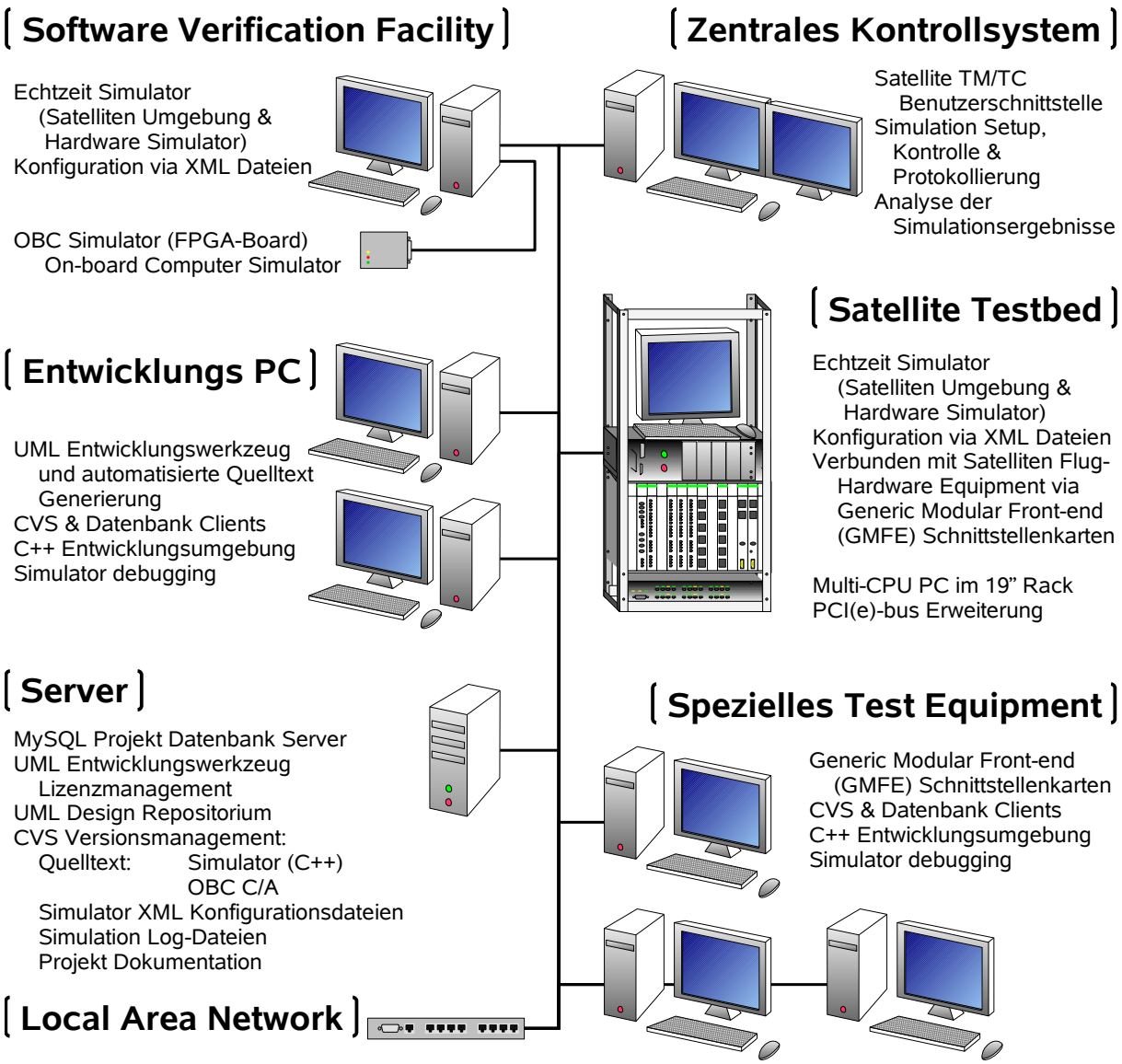


Abbildung 3.2: Graphische Darstellung der Computerhardware-Infrastruktur

Die SVF, das zentrale Missionskontrollsystem und zwei Entwicklungsarbeitsplätze sind normale i586 Standardcomputer. Sie sind im MDVE-Labor untergebracht. Der Echtzeit-Teststand Computer (Satellite Testbed) im Satelliten-Integrationslabor sollte mit mehreren

Prozessorkernen und einem schnellen internen Bussystem ausgestattet sein, um die enorme Rechenleistung von Echtzeit-Simulationen zu bewerkstelligen. Eine Reihe von PCI-Bus Steckkartenplätze und ein 19“ Rackgehäuse sind ebenfalls nötig. Schließlich übernehmen drei weitere Standard i586 Computer mit mehreren freien PCI-Bus Steckkartenplätzen die Aufgabe des Special Check-out Equipments (SCOE). Zur vereinfachten administrativen Softwarewartung ist es empfehlenswert, alle Standardcomputer mit der identischen Hardwarekonfiguration auszustatten.

3.1.2 Grundlegende Software Installation

Grundsätzlich werden alle Computer mit dem Betriebssystem SuSE Linux 9.3 professional betrieben. Hauptgründe für diese Wahl sind auf der einen Seite die Stabilität und die weite Verbreitung in größeren Unternehmen, welches generell zu einem großen stabilen Funktionsumfang führt. Auf der anderen Seite ist diese Linux Distribution inklusive Online Updates frei und kostenlos verfügbar, solange kein direkter und individueller Support benötigt wird. Dieses Release ist die letzte stabile Version, die die Anforderungen des Echtzeit-Simulators von Astrium an den Linux Kernel stellt. Tabelle 3.1 unten enthält eine Zusammenfassung der Standard Softwareinstallation auf den Computern im MDVE-Labor.

Die GNU Compiler Collection (GCC) [GCC '85] fungiert als C++ Compiler und der GNU Project Debugger (GDB) [GDB '86] bietet hilfreiche Unterstützung bei der Fehlerbeseitigung. Beide Werkzeuge basieren auf der Open Source Philosophie und sind unter den Bedingungen der GNU GPL lizenziert.

MySQL [MySQL '96] Client oder Server Software Pakete sind auf alle Computern installiert, damit die Projektdatenbank mit ihren Konfigurationsdaten überall verfügbar ist. Während der Server mit MySQL in der Version 5.0 arbeitet, benutzen die Clients die in ihren Linux Distributionen verfügbare Version.

Die quelloffene universelle Entwicklungsplattform Eclipse [Eclipse '01] ist eine Integrierte Entwicklungsumgebung (IDE) und mit den C/C++ Entwicklungswerkzeugen (CDT) bietet es eine umfassende C++ Programmieroberfläche. Eclipse ist ein Nutzer-Frontend und integriert GCC und GDB direkt in die Graphische Benutzeroberfläche, wodurch die Produktivität deutlich erhöht wird.

OpenOffice.org – die freie Office Suite [OOo '02] wird für die gesamte Projektdokumentation benutzt, im Allgemeinen bedeutet dies zwei Dokumente pro modellierter Satellitenkomponente. Am Institut existieren für verschiedene Zwecke projektspezifische Dokumentvorlagen, die dazu genutzt werden können. Weiterhin bietet OpenOffice.org ein graphisches Nutzer-Frontend für die MySQL Projektdatenbank. Existierende Daten können angesehen, hinzugefügt, geändert oder gelöscht werden und umfangreiche Datenbankreports können in vorbereiteten Dokumenten automatisch generiert werden. OpenOffice.org nutzt die GNU Lesser General Public License (LGPL).

Nur das UML Entwicklungswerkzeug Ameos, das von der Firma Aonix GmbH [Aonix '06] vertrieben wird, benötigt eine kommerzielle Lizenz. Astrium verwendet Ameos zur graphischen und abstrakten Programmierung der Simulatorumgebung und der Hardwaremodelle in UML. Anschließend wird mit einem Codegenerator aus den UML-Diagrammen der Quelltext erzeugt. Drei Ameos Server Lizenzen (Linux86) stehen am Institut für Raumfahrtsysteme zur Verfügung und können von den Entwicklungsarbeitsplätzen aus aktiviert werden, um sowohl UML Diagramme für Hardwaremodelle zu erzeugen als auch den Codegenerator zu betreiben. Anfang 2007 entschied sich die Aonix GmbH den Ameos

Quelltext unter leicht modifizierten Bedingungen der LGPL zu veröffentlichen und mit OpenAmeos [ScopeSET '07] ein Open Source Projekt zu starten. OpenAmeos repräsentierte zu Beginn eine „saubere“ Version des vorherigen kommerziellen Produkts. Leider existiert bisher noch keine Linux basierte Version und daher wird am Institut für Raumfahrtsysteme bis auf Weiteres noch mit der letzten kommerziellen Version gearbeitet.

Tabelle 3.1: Benötigte und installierte Softwarepakete auf den MDVE-Labor Computern

Software Paket	Aufgabe	Betreffende Computer	Lizenzanzahl / -typ
SuSE Linux 9.3 professional	Betriebssystem	alle	Kostenlos verfügbar
gcc, gdb, cvs	Software Entwicklung	alle	GPL
MySQL	Projekt Datenbank	Server & Clients	GPL
Eclipse	Integrierte Entwicklungsumgebung	alle	EPL (Eclipse Public License)
OpenOffice.org	Dokumentation, Datenbank Client	alle	LGPL
Aonix Ameos	UML Entwicklungswerkzeug	Server, Entwicklungsarbeitsplätze	3 Server Lizenzen, kommerziell
OpenAmeos	quelloffenes UML Entwicklungswerkzeug	alle	LGPL

Eine bedeutende Verringerung des administrativen Aufwands der Computer bringt die Anwendung eines Festplatten-Backup und Wiederherstellungswerkzeugs. Es erlaubt das Wiederherstellen eines komplett installiertem Systems auf einem anderen Computer. Inklusive kleinerer Änderungen des Computernamens (Hostname) und der Netzwerkkarteneinstellungen dauert der ganze Prozess nicht mehr als zwanzig Minuten. Allerdings erfordert dieses Prinzip, dass alle beteiligten Computer die identische Hardwareausstattung besitzen, welches bereits bei der Beschaffung der Computer berücksichtigt wurde. Partimage [Partimage '00] ist Element der SystemRescueCd [SysRescCd '03] und ein empfehlenswertes quelloffenes Werkzeug, um die oben genannten Funktionen zu erfüllen.

3.1.3 Das *Flying Laptop* Concurrent Versions System Repositorium

In der Natur der Programmierung von größeren Software-Projekten mit mehreren Entwicklern liegen häufige Änderungen, Umstrukturierungen, Erweiterungen und Fehlerbeseitigungen. Eine große Hilfe bei der Nachverfolgung von Änderungen im Quelltext ist das Concurrent Versions System (siehe Kapitel 2.4.1), das daher auch im *Flying Laptop* Projekt angewendet wird. Es erlaubt das Wiederherstellen und Einsehen von älteren Dateiversionen und stellt Informationen bereit, wer wann etwas geändert hat. Zentral auf einem Server installiert, wird somit auch ein Datensicherungsprinzip einfach realisiert.

Im *Flying Laptop* Projekt wird das CVS für die Versionierung und die Nachverfolgung von Quelltexten und Dokumenten eingesetzt. Bei der Verwendung für Dokumente, die in der Regel als binäre Dateien vorliegen, verliert das CVS den Vorteil der Darstellung der einzelnen

Änderungen, jedoch dient es allen Team-Mitgliedern als automatisches Meldungssystem, wer wann welches Dokument geändert hat. Die inhaltlichen Änderungen können dann über die Änderungsverfolgung im Dokument selbst dargestellt werden.

Das auf dem MDVE Server installierte CVS stellt dem Projekt Team Repositorien für den Simulator-Quelltext und von Astrium bereitgestellte Binärdateien zur Verfügung. Weiterhin gibt es ein Repositorium für die gemeinsame Entwicklung der Projektdokumentation und dem Quelltext für die On-board Computer Algorithmen. Die Verwendung von unterschiedlichen Repositorien ermöglicht die Einschränkung von Zugriffsrechten bestimmter Benutzergruppen.

3.1.4 Die *Flying Laptop* Projekt Datenbank

Das Datenbanksystem des *Flying Laptop* Projekts erfüllt zwei wesentliche Aufgaben: Zum Einen die Unterstützung des Projektmanagements durch Datentabellen wie die Auflistung der Dokumentations- und Zeichnungsnummern und eine Sammlung von hardwarespezifischen Parametern und Informationen. Zum Anderen enthält die Datenbank eine Vielzahl von verknüpften Tabellen für die Konfiguration des Systemsimulators. Dazu gehören in erster Linie zum Beispiel die Konfigurationsparameter der Modelle und ihrer Instanzen im Simulator, die Parameter für die Simulatorkonfiguration selbst und schließlich die datenbankbasierte Generierung von Testfällen.

Die Datenbank ist auf dem MDVE Server installiert und ist damit auch Teil der täglichen Datensicherung. Jeder Benutzer hat einen eigenen Account und kann entsprechend seinen Zugriffsrechten Daten einsehen, ändern oder ergänzen. Für die Arbeit mit der Datenbank wird die OpenOffice.org Komponente `Base` verwendet. Sie ermöglicht die direkte Arbeit mit den Tabellen oder aber die Datenbearbeitung mit Hilfe von Formularen. In Reports lassen sich die Daten übersichtlich zusammenfassen, beliebig formatieren und in andere Dokumente übertragen.

3.1.5 Automatisierte Erzeugung der Konfigurationsdateien des Simulators

Zum Simulationsstart werden alle Komponenten des Systemsimulators mit geeigneten Parametern initialisiert. Diese Parameter sind in XML-Konfigurationsdateien einer vordefinierten Struktur entsprechend abgelegt. In der Projekt Datenbank sind alle Basisparameter und Testfall-abhängige Parameter definiert. Je nachdem welcher Test simuliert werden soll, muss ein eigenständiger konsistenter Satz von Konfigurationsdateien aus der Datenbank generiert werden.

Je mehr Modelle an der Simulation teilnehmen, desto größer und komplexer werden die Konfigurationsdateien. Das manuelle Ändern der Dateien ist sehr aufwendig und geschieht immer mit dem Risiko von Fehlern. Daher übernimmt die Aufgabe der Testfall-abhängigen automatisierten Generation von Konfigurationsdateien das kleine Software Werkzeug `mysql2xml`, welches speziell zu diesem Zweck entwickelt wurde. Es wird im Weiteren detaillierter beschrieben.

Die Entscheidung unter mehreren Möglichkeiten ist auf ein kleines eigenständiges C++ Kommandozeilenprogramm gefallen, welches die Datenbank nach den Einträgen der Dateien mittels einer MySQL API abfragt und sowohl gültige XML Konfigurationsdateien als auch die jeweils zugehörige DTD erzeugt. Das Programm `mysql2xml` erfüllt die folgenden Funktionen:

- Schnelle Erzeugung von gültigen XML Konfigurations- und den zugehörigen DTD Dateien.
- Es können entweder alle Dateien oder die per Parameter spezifizierte Datei bei einem Aufruf erzeugt werden. Es wird immer paarweise die XML Datei und die zugehörige DTD Datei erzeugt.
- Per Parameter kann der Ausgabeort der Dateien definiert werden.
- Die zu generierenden Testfall-abhängigen Konfigurationsdateien werden über einen gesonderten Parameter, den so genannten „Test Case“ definiert.
- Das Programm kann per Parameter im Stapelverarbeitungsmodus gestartet werden. In diesem Fall gibt das Programm beim Beenden einen Statuscode zurück, der über Erfolg bzw. Misserfolg der Programmausführung informiert. Diese Option dient dem späteren Anbinden von `mysql2xml` an eine Graphische Benutzeroberfläche.
- Eine Konfigurationsdatei mit den Parametern der IP Adresse, dem MySQL Benutzer und dem Namen der Datenbank spezifiziert das betreffende Datenbanksystem, an das die Anfragen gesendet werden sollen.
- Eine Benutzerhilfe bzw. eine Anleitung wird auf Anforderung oder immer dann angezeigt, wenn das Programm mit falschen Parametern aufgerufen wurde.

3.2 Die Software Verification Facility und die Anwendung im *Flying Laptop* Projekt

Der Fokus in diesem Kapitel liegt auf den organisatorischen Aspekten der Anwendung der modell-basierten Entwicklungs- und Verifikationstechnologie innerhalb eines Kleinsatellitenprojektes. Im Rahmen dieser Dissertation wird im Speziellen auf den Aufbau der Software-Verifikations-Einrichtung, einer Entwicklungs- und Testplattform für die On-board Kontrollalgorithmen eingegangen. Im *Flying Laptop* Projekt soll der MDVE soweit möglich für die folgenden Aufgaben zum Einsatz kommen:

- Entwicklung und Verifikation der On-board Computer Kontrollalgorithmen
- Entwicklung und Validierung der Kontrollalgorithmen für die Lageregelung des Satelliten
- Flugprozedur Entwicklung und Validation
- Validierung von Kommandosequenzen zwischen Missionskontrollsystem und Satellit und zwischen On-board Computer und der einzelnen Satellitenhardware
- Validierung der Hardware $\leftrightarrow$ Software Kompatibilität
- Testumgebung für die Satellitenflughardware (systemweit bzw. so weit wie möglich)

Die Erfüllung dieser Aufgaben erfordert unterschiedliche MDVE Teststände, die in den folgenden Absätzen identifiziert und erklärt werden. Es gibt mehrere unterschiedliche Entwicklungsstufen der MDVE Simulationsumgebung, die jeweils speziellen Aufgaben dediziert werden können.

Zu den im Rahmen dieser Dissertation am Institut für Raumfahrtsysteme entwickelten Neuerungen zählt die Einbindung eines FPGA-basierten On-board Computers in den Simulationskreislauf als Hardware-in-the-loop. Der Ansatz einer rein softwarebasierten Simulationsumgebung inklusive der realen für einen FPGA geschriebenen On-board Software stellte sich als nicht praktikabel heraus. Statt dessen wurde die On-board Software auf einem FPGA-Entwicklungsboard ausgeführt und über ein Simulator-Frontend mit der Systemsimulation verbunden. Näheres dazu ist in Kapitel 3.3.1 erläutert.

3.2.1 Generelles zur Implementation des Systemsimulators am Institut für Raumfahrtsysteme der Universität Stuttgart

Dank einer intensiven Kooperation zwischen der Firma EADS Astrium GmbH in Friedrichshafen am Bodensee und dem Institut für Raumfahrtsysteme der Universität Stuttgart ist es dem universitären Kleinsatellitenprojekten möglich, beim Aufbau einer Systemsimulationsumgebung auf professionelle Unterstützung zurückzugreifen. Dies bezieht sich nicht nur auf Informationen und systemkritische Diskussionen, sondern auch auf die Bereitstellung von wesentlichen Simulatorkomponenten von Seiten der EADS Astrium GmbH. So stellt Astrium die essentiellen Kernelemente des Systemsimulators zur Verfügung. Dem Institut obliegen folglich die Aufgaben der Einbindung dieser Elemente in eine Systemsimulationsinfrastruktur und die Erstellung von projekthardwarespezifischen Modellen des jeweiligen Satelliten. Unter der Systemsimulationsinfrastruktur ist sowohl die Hardware wie das MDVE-Labor und alle

beteiligten Computersysteme als auch die notwendigen softwaretechnischen Werkzeuge und Einrichtungen wie das Quelltext Repositorium, die Projekt Datenbank, die automatisierte Simulatorkonfiguration aus der Datenbank und die Einbindung des UML Entwicklungswerkzeugs zu verstehen. Erst durch die Nutzung von aktuellsten Entwicklungstechnologien ist die Erstellung eines Systemsimulators in diesem Umfang möglich.

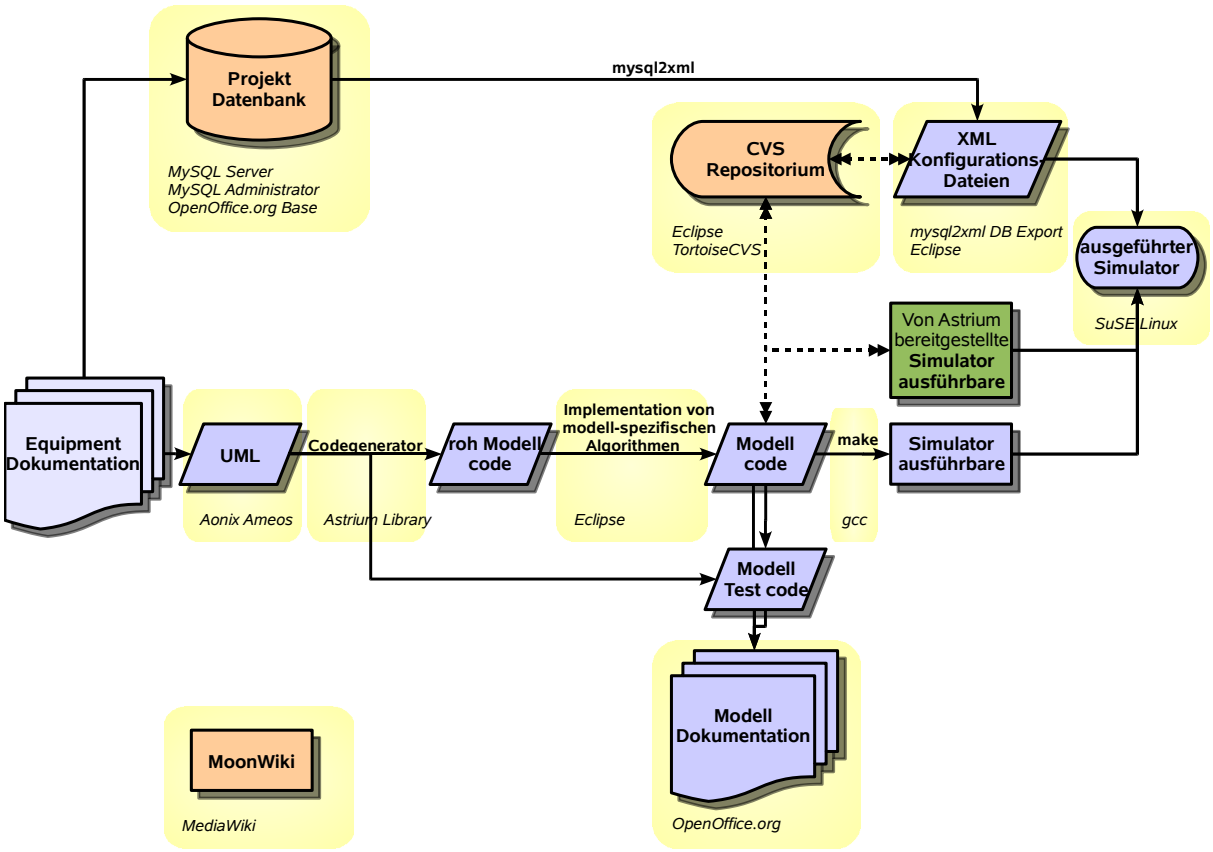


Abbildung 3.3: Ablauf der Erstellung und Integration eines Satelliten Komponentenmodells in den Systemsimulator mit den beteiligten Werkzeugen und Elementen graphisch dargestellt

Am Beispiel der Erstellung eines Satelliten Komponentenmodells lassen sich die benötigten Werkzeuge und Infrastruktur-Einrichtungen am übersichtlichsten erklären. Abbildung 3.3 stellt den Ablauf der Modellerstellung graphisch dar. Aus den links dargestellten Spezifikationsdokumenten des Komponentenherstellers und den eigenen Hardware beschreibenden Dokumenten wird das Konzept für das Modell entwickelt und in UML Diagrammen umgesetzt. Die UML Notation ermöglicht die Vorstrukturierung des Quelltextes, das Anlegen von Methoden und Variablen und das Zuweisen von bestimmten Klassenattributen. Durch den von der Astrium GmbH zur Verfügung gestellten Codegenerator wird im nächsten Schritt der rohe Quelltext aus den UML Diagrammen generiert. Hierbei wertet der Codegenerator alle in der UML Notation vergebenen Attribute aus und fügt entsprechenden Quelltext hinzu. Jetzt kann die modell-spezifische Mathematik und Logik in den Quelltext implementiert und somit die volle Funktionsfähigkeit des Modells hergestellt werden. Mit dem Modellquelltext wird auf der einen Seite weiterer Quelltext zum Testen des Modellquelltextes erzeugt und dokumentiert. Auf der anderen Seite werden aus dem Modellquelltext die ausführbaren Dateien für den lauffähigen Simulator erzeugt. Zusammen mit ausführbaren Dateien des Simulator-Kernels und Subfunktionalitäten, die wiederum von der Astrium GmbH zur Verfügung gestellt wurden, kann der Simulator gestartet werden. Dabei werden zur Simulatorkonfiguration

einige XML-Dateien ausgelesen und die darin enthaltenen Parameter in den Simulator geladen. Die Parameter aus der XML-Datei stammen aus der Projektdatenbank, von wo aus sie mit Hilfe des Datenbank-Extraktionswerkzeuges `mysql2xml` in XML-Dateien vordefinierter Struktur geschrieben werden. Sowohl die Konfigurationsdateien als auch der Modellquelltext steht unter Versionsverwaltung im CVS Repositorium des Projekts. Nicht direkt in den Modellerstellungsablauf eingebunden ist das `MoonWiki`, es stellt jedoch viele kleine Anleitungen und Hilfen für die Modellerstellung und den Umgang mit dem Simulator bereit und wird während des ganzen Vorganges intensiv genutzt.

3.2.2 Definition und Entwicklungsstufen der Systemsimulator Teststände im *Flying Laptop* Projekt

Bei Null beginnend startete die Entwicklung der Simulator-Komponentenmodelle für das *Flying Laptop* Projekt am Institut für Raumfahrtsysteme mit Komponenten des Lageregelungssystems. Da der Weg bis zur vollständigen Software Verification Facility weit ist, wurden insgesamt drei Entwicklungsstufen definiert mit denen bereits erste Simulationen und Tests als Zwischenergebnisse möglich sind. Die Ergebnisse aus diesen ersten Simulation sind ins Projekt zurückgeflossen und führten zu Änderungen und Verbesserungen im Satellitendesign.

Wie bereits mehrfach publiziert ([Falke et al. '06], [Röser et al. '05], [Grillmayer et al. '05a], [Grillmayer et al. '05b]) werden im Folgenden die drei ersten Teststände in ihrer aktuellen Form vorgestellt und detaillierter erläutert.

SVF v0.1

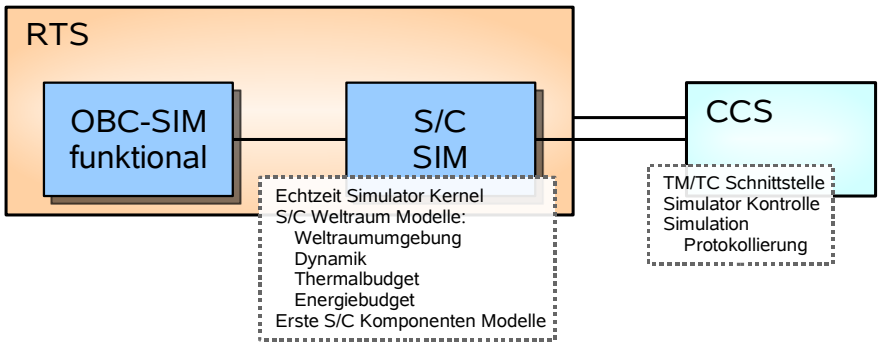


Abbildung 3.4: Graphische Darstellung der Komponenten der Software Verification Facility in der ersten Entwicklungsstufe SVF v0.1

Der erste Entwicklungsschritt des Simulators am Institut für Raumfahrtsysteme wird durch den Gedanken den Simulator „zum Laufen zu bringen“ vorangetrieben. Zu den wichtigsten ersten Schritten dabei zählt in erster Linie der Aufbau der Entwicklungsinfrastruktur (siehe Kapitel 3.2.1 „*Generelles zur Implementation des Systemsimulators am Institut für Raumfahrtsysteme der Universität Stuttgart*“) und die Automatisierung von Simulatorkonfiguration und Start. Repräsentative Modelle der Satellitenhardware stehen hier eher im Hintergrund. Daher ersetzt ein einfaches funktionales Modell für den On-board Computer Simulator eine aufwendigere Repräsentation. Gleichermaßen wird für ein vereinfachtes Modell der Energieversorgungseinheit PCDU verfahren. Bereits vollständig im Echtzeit-Simulator integriert sind allerdings die Weltraummodelle der Simulator-Infrastruktur zur Abbildung der Weltraumumgebung, des dynamischen und des vereinfachten thermalen Verhaltens.

Diese erste Entwicklungsstufe ist eine lauffähige Simulatorkonfiguration, die fehlerlos startet und läuft, so dass für die weitere Entwicklung sukzessive neue Modelle testweise integriert und anschließend fest implementiert werden können. Abbildung 3.4 zeigt die wichtigsten Komponenten dieser Simulatorkonfiguration in graphischer Darstellung während Tabelle 3.2 die Charakteristika dieses Setups umreißt.

Tabelle 3.2: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v0.1

Entwicklungsstufe	■ Systemsimulationsinfrastruktur ist aufgebaut, konfiguriert und unterstützt die Modellentwicklung ■ Integration der Weltraummodelle (Weltraumumgebung, Dynamik, Thermal) des Simulators als Infrastrukturmodelle ■ Einfache Komponentenmodelle ohne Repräsentation der Satelliten-Hardware ■ Der OBC-Simulator wird durch ein funktionales Modell ersetzt ■ Als Zentrales Kontrollsystem dienen die einfachen Textkonsolen des Simulator-Kernels
Integrierte Satelliten Komponenten-Modelle	■ Einfache funktionale Modelle für OBC und PCDU
Ziele, Aufgaben und Fähigkeiten	■ Inbetriebnahme des Systemsimulators am Institut für Raumfahrtsysteme ■ Entwicklungsplattform der detaillierten Komponentenmodelle des Satellitenprojekts *Flying Laptop*

SVF v0.5

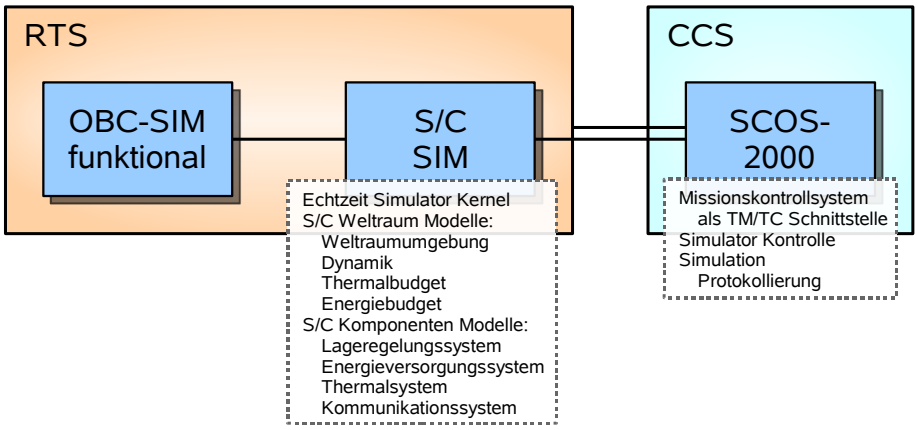


Abbildung 3.5: Graphische Darstellung der Software Verification Facility in der zweiten Entwicklungsstufe SVF v0.5

Im nächsten Entwicklungsschritt der Software Verification Facility liegt der Schwerpunkt auf dem Ziel, bereits erste aussagefähige Simulationen für das Energie- bzw. Thermalverhalten des Satelliten in verschiedenen Betriebsmodi durchführen zu können. Größere strukturelle Änderungen der Simulatorkonfiguration sind nicht erforderlich (siehe Abbildung 3.5), dafür jedoch die Entwicklung einer Vielzahl von Satelliten-Komponentenmodellen im Detail. Die einzige Erweiterung der Simulator-Infrastruktur ist die Integration eines Thermalpropagators zur Beschreibung des detaillierteren thermischen Verhaltens des Satelliten unter Weltraumbedingungen. Im Thermalmodell wird ebenfalls die aktuell als Wärme dissipierte Energie aller Satellitenkomponenten berücksichtigt.

Die Erzeugung von nahezu allen Komponentenmodellen erfolgt mit Hilfe von UML-Diagrammen, einer modell-basierten Entwicklungstechnologie, aus denen durch einen speziellen Codegenerator der Modellquelltext erzeugt wird. Je mehr Modelle entstehen desto umfangreicher wächst auch die Konfigurationsdatenbank. Das Zentrale Kontrollsystem wird durch die Installation, Konfiguration und Inbetriebnahme des Missionskontrollsystems mit umfangreichen Funktionen ausgestattet und dient ab sofort als Schnittstelle zwischen Simulator und Operator. Die Mission Information Base des Missionskontrollsystems enthält erste Definitionen von Telemetrie- und Telekommandopaketen des Satelliten. Die wesentlichen Charakteristika der Software Verification Facility dieser Entwicklungsstufe sind in Tabelle 3.3 unten zusammengefasst.

Tabelle 3.3: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v0.5

Entwicklungsstufe	■ Integration eines Thermalpropagator-Infrastrukturmodells des Simulators ■ Kleine funktionale Erweiterungen des funktionalen OBC-Simulators, die realen On-board Kontrollalgorithmen sind noch nicht implementiert ■ Aufbau und Inbetriebnahme des Missionskontrollsystems
Integrierte Satelliten Komponenten-Modelle	■ Alle Komponentenmodelle des Satelliten, allerdings teilweise mit nur grober Repräsentation der wichtigsten Funktionen
Ziele, Aufgaben und Fähigkeiten	■ Entwicklungsplattform der detaillierten Komponentenmodelle des Satellitenprojekts *Flying Laptop* ■ Erste aussagefähige Simulation für Energie- und Thermalbudget möglich

SVF v1.0

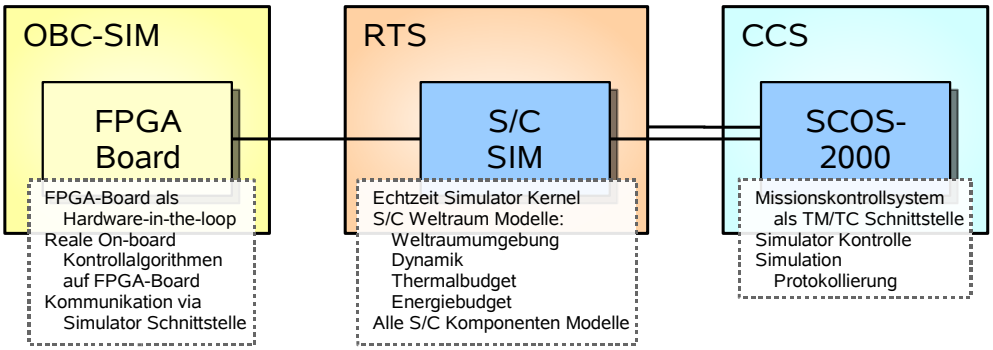


Abbildung 3.6: Graphische Darstellung der Software Verification Facility in der letzten und vollständigen Entwicklungsstufe SVF v1.0

In der vollständigen Entwicklungsstufe der Software Verification Facility ist die wichtigste Neuerung der On-board Computer Simulator (siehe Abbildung 3.6). Dieser ist ein als Hardware-in-the-loop eingebundenes FPGA-Entwicklungsboard und enthält die On-board Kontrollalgorithmen gekoppelt an eine spezielle Simulatorschnittstelle zur Datenübertragung an den Systemsimulator. Dadurch stellt der Systemsimulator eine Entwicklungsplattform für die On-board Kontrollalgorithmen dar und kann mit ihnen erste Simulationen und Tests durchführen. Voraussetzung dafür ist natürlich, dass alle Komponentenmodelle in hohem Detaillierungsgrad abgebildet sind und auch das Missionskontrollsystem voll funktionsfähig ist. Weiterhin ist ein Thermalmodell in erweitertem Funktionsumfang, mit Berücksichtigung von dynamischen Umgebungsbedingungen in Abhängigkeit der relativen Satellitenlage, integriert.

Die sukzessive Entwicklung der On-board Kontrollalgorithmen ermöglicht immer umfangreichere Tests, begonnen von der Kommunikation mit den Modellen der Hardware Komponenten über Tests einzelner Betriebsmodi des Satelliten bzw. des Lageregelungssystems bis hin zu Transitionen zwischen den einzelnen Betriebsmodi. Hier dient der Systemsimulator als Werkzeug für den Funktionsnachweis der On-board Kontrollalgorithmen. Im weiteren Verlauf der Satellitenentwicklung können dann ganze Flugprozeduren entwickelt, getestet und komplette Missionsprofilsimulationen durchgeführt werden. Diese Simulationen können automatisiert und bei Änderungen in den On-board Kontrollalgorithmen jederzeit reproduziert werden. Tabelle 3.4 fasst die wesentlichen Charakteristika der Software Verification Facility dieser letzten Entwicklungsstufe zusammen.

Tabelle 3.4: Charakteristika der Software Verification Facility in Entwicklungsstufe SVF v1.0

Entwicklungsstufe	■ Integration des erweiterten Thermal-Infrastrukturmodells des Simulators ■ Implementation eines OBC-Simulators mit einem FPGA-Entwicklungsboard als Hardware-in-the-loop und den realen On-board Kontrollalgorithmen ■ Missionskontrollsystem voll funktionsfähig
Integrierte Satelliten Komponenten-Modelle	■ Alle Komponentenmodelle des Satelliten in detaillierter Repräsentation der Hardware Charakteristik
Ziele, Aufgaben und Fähigkeiten	■ Entwicklungsplattform für die On-board Kontrollalgorithmen ■ Simulationen und Tests des Lageregelungssystems, Test von Betriebsmodi des Satelliten und Modus Transitionen mit Identifikation von Fehlfunktionen ■ Entwicklung und Test von Flugprozeduren und kompletten Missionsprofilen

3.3 Simulationsmodelle für den *Flying Laptop* System Simulator

Zur vollständigen Simulation des Kleinsatelliten *Flying Laptop* müssen natürlich alle Systemkomponenten des Satelliten (siehe Kapitel 2.1.2 *(Systemkomponenten des Flying Laptop)* und Abbildung 2.3 auf Seite 8) modellhaft im Simulator abgebildet werden. Die Simulationsmodelle sind in Tabelle 3.5 unten aufgeführt und gemäß der im vorherigen Kapitel vorgestellten Simulator Entwicklungsstufen sukzessive erstellt, verfeinert und in den Simulator integriert worden. Bei Einhaltung der vom Autor dieser Dissertation definierten Anforderungen und Randbedingungen erfolgte die Aufgabe der funktionalen Modellierung der Satellitenkomponenten unter Anleitung und Betreuung des Autors weitestgehend im Rahmen der Studien- und Diplomarbeiten von Boris Wyszkowski [Wyszkowski '05], Alexander Brandt [Brandt '07], Ivan Kossev [Kossev '08], Michael Lachenmann [Lachenmann '06], Rainer Saage [Saage '07] und Jens Fischer [Fischer '08].

Tabelle 3.5: Liste der erstellten Simulationsmodelle für den Kleinsatelliten Flying Laptop

Subsystem	Akronym	Simulationsmodell	Anzahl der Instanzen im Satelliten *Flying Laptop*
ACS	RW	Reaktionsrad	4
	MGT	Magnetorquer	2
	MGM	Magnetometer	2
	SuS	Sonnensensorsystem	2
	GPS	GPS Receiver	3
	STR	Star Tracker	1
	FOG	Faseroptische Kreisel	4
PSS	PCDU	Energieverteilungseinheit	1
	SPL/M/R	Solargenerator	1
	BATT	Batteriesystem	1
C&DH	OBC	On-board Computer (Einfaches Model)	1
	OBC	FPGA in-the-loop	1
TT&C	S TX/RX	S-band Transmitter/ Receiver	1/2
	UHF TX/RX	UHF Transmitter/ Receiver	1/2
	VHF TX	VHF Transmitter/ Receiver	1
P/L	KA TX/RX	Ka-band Transmitter/ Receiver	1/1
	KU TX	Ku-band Transmitter	1
	MICS	Optisches Kamerasystem	3
	TICS	Thermal-Infrarot Kamerasystem	1
	PAMCAM	Panorama Kamera	1
TCS		Temperatur Sensoren/ Heizelemente	>50/8

Beispielhaft werden hier zwei vom Autor dieser Dissertation maßgeblich erstellte Modelle näher erläutert. Die Anbindung des FPGA-Entwicklungsboards an den Systemsimulator stellt die wichtigste Neuerung dar und wird hier explizit im Detail erklärt. Im Weiteren wird als anschauliches Beispiel der Simulationsmodelle der Satellitenkomponenten das Solarzellen-gestützte Sonnensensorsystem vorgestellt, um den in den Simulationsmodellen realisierten Detaillierungsgrad zu verdeutlichen.

3.3.1 On-board Computer Simulator mit FPGA Hardware

Der wichtigste Bestandteil einer Systemsimulationsumgebung eines Satellitenprojektes ist die Einbindung der On-board Software. Im Falle des *Flying Laptop's*, mit einem FPGA-basierten On-board Computer System, ist dies die Einbindung der Kontrollalgorithmen des FPGA's. Traditionelle On-board Computer können bei ausreichend Rechenleistung des Simulators über Prozessoremulationen erfolgreich simuliert werden. Der gleiche Ansatz ist jedoch für einen FPGA-basiertem On-board Computer nicht möglich. Die Simulation eines solchen On-board Computers ist bisher noch nicht versucht worden und stellt nicht nur im Kontext einer Kleinsatellitenmission eine Neuerung dar. Dementsprechend konnte auf keinerlei Erfahrung oder mögliche Ansätzen zurückgegriffen werden.

Mit Unterstützung ihrer Betreuer hat Viola Wolter in ihrer Diplomarbeit [Wolter '06] eine Machbarkeitsanalyse durchgeführt, inwieweit sich die für den FPGA programmierten Kontrollalgorithmen in den Systemsimulator einbinden lassen. Insgesamt zeigte sich, dass es zwar Möglichkeiten der Simulation von FPGA programmiertem Quelltext in Form von Interpretern der Programmiersprache oder aber bereits vorgefertigten Simulationswerkzeugen gibt, aber dass dabei unter anderem eine Verlangsamung der Simulationsgeschwindigkeit um den Faktor 500 oder mehr im Verhältnis zur Echtzeit-Simulation erkauft wird. Für Zwecke der Systemsimulation im Kontext einer Satellitenmission ist diese Verlangsamung nicht tragbar. Es musste eine andere Lösung gefunden werden.

Prinzipiell dient die Einbindung des On-board Computers in die Systemsimulation zwei Zielen: Erstens der Entwicklung und funktionellen Überprüfung der Kontrollalgorithmen und zweitens der Verifikation des gesamten On-board Computer Systems inklusive der darin befindlichen Kontrollalgorithmen. Der Fortschritt im *Flying Laptop* Projekt legte vorerst einen Schwerpunkt auf das erste Ziel der Entwicklung und funktionalen Überprüfung der Kontrollalgorithmen nahe, da der On-board Computer als Ganzes noch nicht vorhanden war.

Zur Realisierung dieser Aufgabe im Rahmen der Systemsimulation wurde entschieden, ein FPGA-Entwicklungsboard als Hardware in die Simulationsumgebung einzubinden und auf diesem Entwicklungsboard die realen On-board Kontrollalgorithmen zu betreiben. Weiterhin sollen alle Datenleitungen zu den einzelnen Satellitenkomponenten über eine einzige Verbindung in den Systemsimulator zu den simulierten Komponentenmodellen gebündelt werden. Diese strukturelle Änderung vereinfacht die Anbindung des Entwicklungsboards an den Simulator erheblich, erfordert aber ein spezielles Simulator-Frontend in den On-board Kontrollalgorithmen (siehe Abbildung 3.7 unten). Durch dieses Simulator-Frontend erfolgt die Datenübertragung nicht direkt an die eigentlich parallel zueinander angeschlossene Satellitenhardware, sondern das inhaltlich unveränderte Datenpaket wird zum Systemsimulator gesendet und dort intern zum entsprechenden Komponentenmodell weitergeleitet. Auf der Seite des Systemsimulators dient eine spezielle Schnittstellenkarte (Generic Modular Frontend, GMFE) dem Empfang und der Weiterleitung der Daten an den Systemsimulator. Dazu ist ein dedizierter Schnittstellenkarten-Treiber als separate Anwendung programmiert worden. Dieser realisiert den Datenaustausch zwischen der Schnittstellenkarte und dem Frontend-

Modell `OBCdev_FE` des Systemsimulators in beide Richtungen durch einen gemeinsamen Speicherbereich und ermöglicht die entsprechende interne Weiterleitung. Somit ist eine klare Trennung zwischen dem Systemsimulator und dem Schnittstellenkarten-Treiber als Betreiber der Hardware gewährleistet. Der Rückweg der Kommunikation vom Komponentenmodell zum Simulator-Frontend und zu den On-board Kontrollalgorithmen erfolgt über den gleichen Weg.

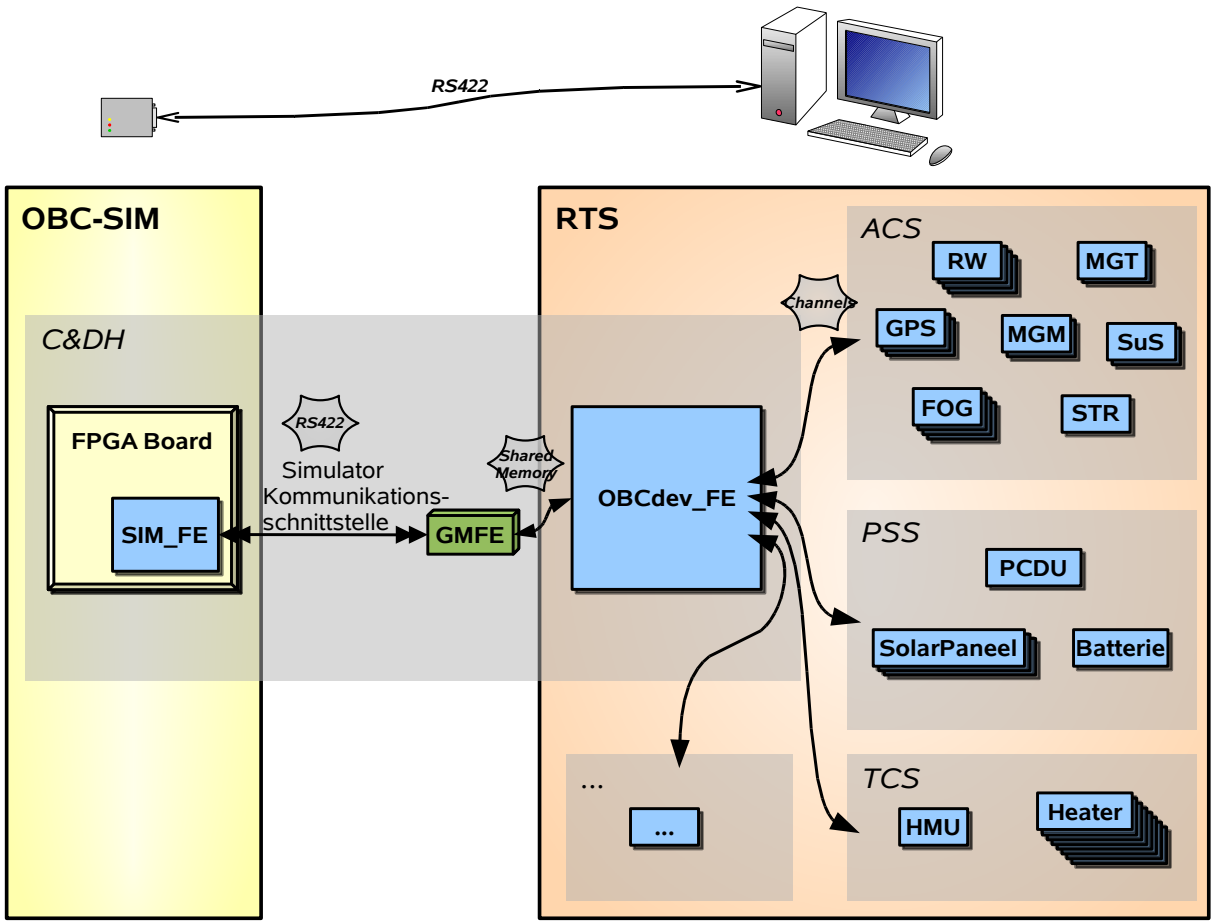


Abbildung 3.7: Schematische Struktur der Entwicklungs Software Verification Facility mit Echtzeit-Simulator (RTS) und implementiertem On-board Computer Simulator (OBC-SIM)

Zur Vermeidung von Wartezeiten bzw. Verspätungen der Daten aufgrund der Datenbündelung durch die einzelne Schnittstelle zum Systemsimulator ist die digitale schnelle serielle Schnittstelle RS422 (Recommended Standard 422) gewählt worden. Sie wird bidirektional mit der Übertragungsrate von 921600 Baud betrieben. Weitere Details zur konkreten Umsetzung des hier vorgestellten On-board Computer Simulators werden in den folgenden Abschnitten erläutert.

Von Seiten der Systemsimulation wird eine Schnittstelle zur Anbindung eines FPGA-Entwicklungsboards bzw. der On-board Kontrollalgorithmen an den Systemsimulator zur Verfügung gestellt. Das Gegenstück zur Simulatorschnittstelle, das Simulator-Frontend (`SIM_FE`) wird von den Entwicklern der Kontrollalgorithmen bereitgestellt. Ziel der Unternehmung ist eine Entwicklungsplattform für die Kontrollalgorithmen des On-board Computers zu schaffen und somit bereits zum jetzigen Entwicklungszeitpunkt Systemsimulationen durchführen zu können. Somit lässt sich die Funktionalität der Kontrollalgorithmen verifizieren.

3.3.1.1 Datenübertragung über die serielle Schnittstelle

Aufgrund der einfachen und bereits vom Hersteller vorbereiteten Implementierung der seriellen Schnittstelle auf dem FPGA-Entwicklungsboard ist die RS422 als Schnittstelle zur Datenübertragung gewählt worden. Die ersten Versuche wurden mit der RS232 durchgeführt, woraufhin der Ausbau zur RS422 mit höherer Datenübertragungsrate von 921600 Baud erfolgte. Ein wesentlicher Vorteil der RS422 gegenüber anderen heute üblichen Datenübertragungsschnittstellen für Peripheriegeräte (z. B.: USB, Ethernet) ist die dedizierte bidirektionale Punkt zu Punkt Datenübertragung mit hoher Übertragungsrate und ohne Latenzzeiten auf dem Datenbus.

Während der Simulationscomputer mit einer kommerziellen RS422 Schnittstellenkarte ausgestattet wurde, ist auf dem FPGA-Entwicklungsboard der RS232 Mikrocontroller mit einem RS422 Mikrocontroller ausgetauscht worden. Die weiteren Einstellungen der Schnittstelle erfolgten dann durch den jeweiligen Treiber der Schnittstellenkarte. Eine explizite Datenflusskontrolle kommt nicht zum Einsatz, weder hardware- noch softwaregesteuert. Statt dessen werden alle Daten in ein definiertes Datenpaket verpackt, den so genannten Simulator-Transfer-Frame (STF), der im Folgenden Abschnitt detaillierter erläutert wird. Die jeweiligen Schnittstellen-Treiber der Anwendungen übernehmen die Kontrolle über das Senden und Empfangen von jeweils vollständigen Datenpaketen.

Datenübertragung in Simulator-Transfer-Frames

Zur Übertragung von Datenpaketen zwischen FPGA-Entwicklungsboard und den Komponentenmodellen werden die Pakete in Simulator-Transfer-Frames verpackt. Der Aufbau eines Simulator-Transfer-Frames ist in Abbildung 3.8 unten ersichtlich.

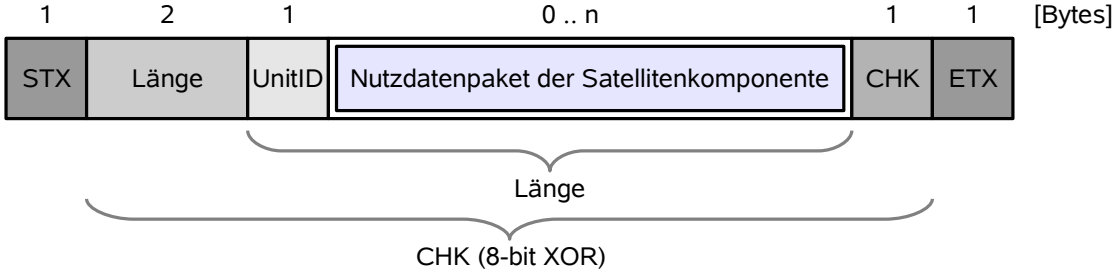


Abbildung 3.8: Strukturdefinition des Simulator-Transfer-Frames

Das `STX` (`0x02`, Start Transmission) Byte und das `ETX` (`0x03`, End Transmission) Byte sind für die implementierte Form der Flusskontrolle der seriellen Schnittstelle erforderlich. Sie ermöglichen das exakte Erkennen von Anfang und Ende der zu übertragenden Datenpakete.

Die Längeninformation im Feld `Länge` muss entsprechend der Länge des Datenpakets der Satellitenkomponente zuzüglich einem Byte für das Feld `UnitID` gesetzt werden. Das Nutzdatenpaket der Satellitenkomponente kann die Länge Null haben, um spezielle mit einem Byte kodierte Nachrichten zu versenden. Diese Nachrichten werden dann im Feld `UnitID` eindeutig spezifiziert. Die Länge wird in der “Little-Endian-Byte-Order” übertragen, d. h. das niederwertigste Byte (Least Significant Byte, LSB) wird zuerst übertragen. Die Felder `STX`, `ETX` und `CHK` werden in der Längeninformation nicht mitgezählt.

Das Feld `UnitID` enthält die Quell- bzw. Zielinformation des jeweiligen Nutzdatenpakets der Satellitenkomponente. Jeder Satellitenkomponente ist eine einmalige Nummer zugeordnet, anhand derer sie zweifelsfrei identifiziert werden kann. In der Umgebung des Programmquelltextes ist es hilfreich ein Prinzip zu verwenden, dass es ermöglicht die Zuordnung einer Satellitenkomponente zu einer Identifikationsnummer zu ändern, ohne den Quelltext ändern zu müssen. Dies kann mit Hilfe des `ENUM` Datentyps realisiert werden, der `Integer` Zahlen zu Symbolen zuordnet. Nutzt man anschließend diese Symbole wie `SUS0`, `SUS1` oder `FOG3` im Quelltext, so werden die Symbole beim Kompilieren des Quelltexts durch die entsprechenden Integer Zahlen ersetzt. Der hier verwendete `ENUM` Datentyp ist einer gemeinsamen Header Datei `UnitID.h` definiert worden und wird sowohl vom Systemsimulator als auch von Seiten der On-board Kontrollalgorithmen in den Quelltext eingebunden.

Die Checksumme im Feld `CHK` wird aus allen Bytes in den Kopfbereich (`Länge`, `UnitID`) und dem Nutzdatenpaket des Simulator-Transfer-Frames berechnet. Der Algorithmus ist eine 8-bit Addition ohne Übertrag (Exklusives Oder, XOR). Wenn der Frame korrekt übertragen worden ist, muss die XOR Addition aller Bytes und die übertragene Checksumme im Feld `CHK` per Exklusivem Oder addiert Null ergeben.

Die Feldergrößen des Simulator-Transfer-Frame Formats sind definiert wie in Abbildung 3.8 aufgeführt. Die gegebene Länge im Feld `Länge` muss mindestens ein Byte groß sein, da für das Datenpaket der Satellitenkomponente eine minimale Länge von Null Bytes erlaubt ist und das Feld `UnitID` auf jeden Fall gezählt wird. Daraus folgt eine minimale erlaubte Länge für den gesamten Simulator-Transfer-Frame von sechs Bytes. Diese Definitionen sind in Tabelle 3.6 zusammengefasst.

Tabelle 3.6: Definition der Feldlängen des Simulator-Transfer-Frames

Name des Feldes	**Definierte Länge in [Bytes]**
`STX`	1
`Länge`	2
`UnitID`	1
Datenpaket der Satellitenkomponente	0 .. n
`CHK`	1
`ETX`	1
Minimale Länge des Simulator-Transfer-Frames	6

Überprüfung der korrekten Datenübertragung

Zur Überprüfung der korrekten Datenübertragung dient die im Simulator-Transfer-Frame mit übertragene Checksumme im Feld `CHK`. Sie muss dazu gleich aller empfangenen und in der Checksumme berücksichtigten Bytes sein. Der Test auf Korrektheit des Datenpakets erfolgt in der nachfolgenden Reihenfolge. Wenn einer der Tests fehlschlägt, wird der gesamte übertragene Simulator-Transfer-Frame als fehlerhaft identifiziert und inhaltlich ignoriert, d. h. er wird nicht weiterverarbeitet und verworfen.

- Zuerst wird die Länge der empfangenen Daten überprüft. Sie muss mindestens die Länge des kleinsten möglichen Simulator-Transfer-Frames haben, d. h. sie muss nach Tabelle 3.6 mindestens 6 Bytes groß sein.
- Im zweiten Schritt wird die übertragene Länge im Feld `Länge` überprüft. Sie muss mindestens 1 Byte angeben, das Byte für das Feld `UnitID`.
- Schließlich wird die Checksumme durch den Vergleich der übertragenen und der berechneten Checksumme des Simulator-Transfer-Frames überprüft.

Kontrollframes zur Übermittlung spezieller Informationen

Die eigens definierten Kontrollframes sind kurze Nachrichten an den Empfänger, um diesen über ein geschehenes Ereignis, eine erledigte Aufgabe zu informieren oder eine bestimmte Reaktion am Empfänger auszulösen. Kontrollframes können sowohl vom Systemsimulator als auch vom FPGA-Entwicklungsboard gesendet werden. In der Regel enthalten die Kontrollframes kein Nutzdatenpaket, sie sind nur durch die im Feld `UnitID` übertragene Kennnummer definiert. Auch die Kennnummer ist durch den `ENUM` Datentyp (basierend in der gemeinsamen Definitionsdatei) festgelegt. Die nachfolgende Tabelle 3.7 listet die definierten Kontrollframes auf.

Tabelle 3.7: Liste der definierten Kontrollframes

Symbol des Kontrollframes	Kontrollframe Name	Beschreibung
`CF_FPGA_RESTART`	FPGA Restart	Setzt das FPGA-Entwicklungsboard auf den Anfangszustand zurück bzw. startet es neu
`CF_LAST_ITEM`	Last Item	Wird bei sequentieller Ablaufsteuerung als letztes Datenpaket einer Sequenz übertragen
`parameter_dump`	Parameter Zustand	Kennzeichnet ein Datenpaket mit einer Vielzahl von darin übertragenen Parameterwerten

Die Funktionalität der Kontrollframes muss auf beiden Seiten der Datenkommunikation implementiert sein, um die korrekte Interpretation und Reaktion zu gewährleisten.

Das DLE Protokoll

Zur Übertragung von beliebigen Datenpaketen muss ein Protokoll zur Datenübertragungsumschaltung (Data Link Escape, DLE) angewandt werden. Da die speziellen Bytes `0x02` und `0x03` bereits Sonderaufgaben im Rahmen der Definition des Simulator-Transfer-Frames übernehmen, dürfen diese Werte nicht mehr innerhalb des übertragenen Datenpakets auftauchen, da es sonst zu Fehlinterpretationen kommen würde. Das Gleiche gilt für die Steuerzeichen `0x0D` und `0x10`. Das hier angewandte Verfahren ersetzt ein solches auftauchendes Byte innerhalb des Datenpakets durch eine eindeutige Zwei-Byte Sequenz, so dass die einzigen verbleibenden Steuerzeichen innerhalb eines Simulator-Transfer-Frames das erste und letzte Byte sind.

Die Vorwärts- und Rückwärts-Ersetzung vor der Datenübertragung bzw. nach dem Datenempfang wird demzufolge auf alle Bytes zwischen dem Startbyte `STX` und dem Endbyte `ETX` angewandt. Tabelle 3.8 definiert in einer Ersetzungstabelle die Zuordnung von Steuerzeichen mit ihrer entsprechenden Ersetzungssequenz.

Tabelle 3.8: Steuerzeichen Ersetzungstabelle des DLE Protokolls

Steuerzeichen	**entsprechende Ersetzungssequenz**
`0x02 (STX)`	`0x10 0x42`
`0x03 (ETX)`	`0x10 0x43`
`0x0D (CR)`	`0x10 0x4D`
`0x10 (DLE)`	`0x10 0x10`

Wird innerhalb eines empfangenen Simulator-Transfer-Frames ein DLE Steuerzeichen `0x10` erkannt und das nachfolgende Zeichen entspricht nicht einer der definierten Ersetzungssequenz, dann wird das ganze Paket als Fehlerhaft übermittelt betrachtet und verworfen, d.h es wird nicht weiter ausgewertet.

3.3.1.2 Flusskontrolle der Daten und Synchronisation

Die Flusskontrolle der Daten der beteiligten Kommunikationsteilnehmer bzw. die Synchronisation der beiden parallel laufenden Computersysteme spielt eine wichtige Rolle bei der durchgeführten Simulation unter Echtzeit-Bedingungen. Die On-board Kontrollalgorithmen des Lageregelungssystems als maßgebliche Elemente werden mit einem Takt von 10 Hz ausgeführt, d. h. sie werden alle 100 ms basierend auf den zu diesem Zeitpunkt vorliegenden Informationen erneut ausgeführt. In diesem Zusammenhang ergeben sich zwei unterschiedliche Prinzipien zur Flusskontrolle der Daten und Synchronisation. Die sequenzielle und die parallele Ausführung der beteiligten Elemente. Während bei der sequenziellen Ablaufsteuerung die wechselweise Ausführung gewährleistet werden muss, folgt für den parallelen Ablauf, dass die Datenkommunikation jeweils innerhalb der 100 ms Intervalle vollständig abgeschlossen sein muss. Gleichzeitig muss eine explizite Synchronisation der beiden beteiligten Computersystem implementiert werden. Die beiden Prinzipien werden in den folgenden Absätzen kurz vorgestellt. Sie sind nacheinander für den Systemsimulator im Kleinsatellitenprojekt *Flying Laptop* umgesetzt worden.

Sequenzielle Ablaufsteuerung

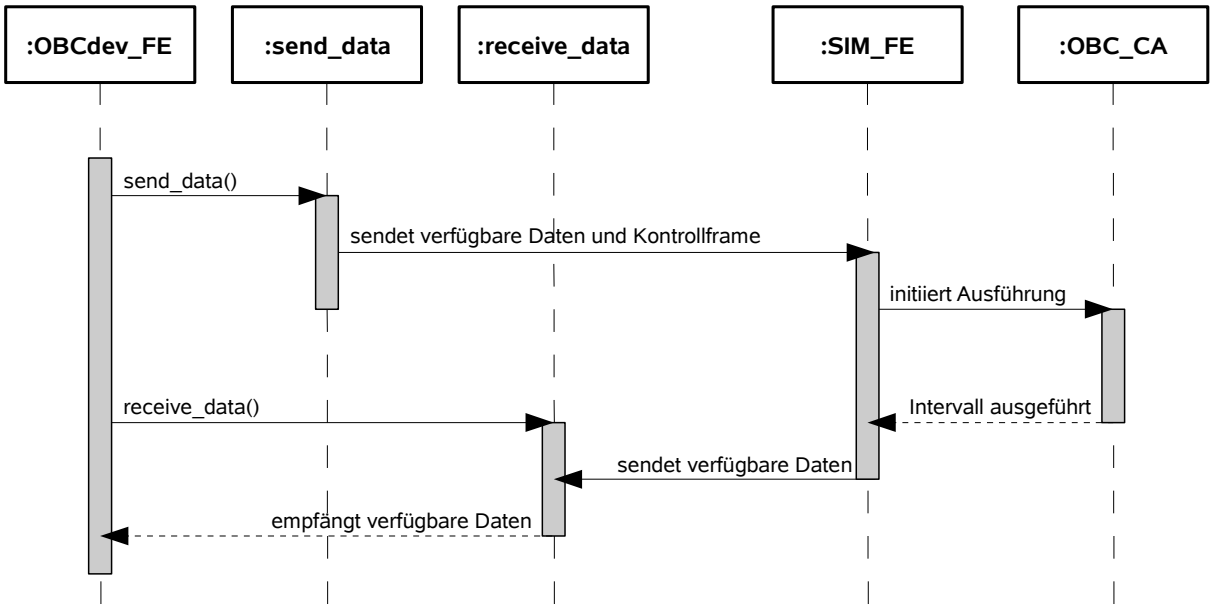


Abbildung 3.9: UML Sequenzdiagramm der Sequentiellen Ablaufsteuerung

Ein explizit sequenzieller Ablauf der in der Systemsimulation beteiligten Elemente wird durch eine Token-basierte Synchronisation gekennzeichnet. Gemeint ist damit die aufeinander folgende und wechselweise Abarbeitung der beteiligten Algorithmen, gesteuert durch die Weitergabe eines Tokens. In dieser Konfiguration ist der Systemsimulator das Hauptsteuerorgan, während das FPGA-Entwicklungsboard die untergeordnete Rolle übernimmt. Wie in Abbildung 3.9 ersichtlich, wird jede Ausführung der On-board Algorithmen des FPGA-Entwicklungsboards vom Systemsimulator initiiert.

In der sequentiellen Ablaufsteuerung ist die Flusskontrolle der Daten wie folgt realisiert:

- Im Frontend-Modell `OBCdev_FE` des Systemsimulators werden alle zum aktuellen Taktzyklus verfügbaren Datenpakete der Satellitenkomponentenmodelle eingelesen, in Simulator-Transfer-Frames verpackt und über den Prozess `send_data` zum FPGA-Entwicklungsboard gesendet. Anschließend wird der Kontrollframe `CF_LAST_ITEM` gesendet und der Simulator begibt sich im Empfangszustand `receive_data` in Wartestellung.
- Das Simulator-Frontend `SIM_FE` auf dem FPGA-Entwicklungsboard ist in Wartestellung und empfängt nacheinander alle ankommenden Daten. Die einzelnen Nutzdatenpakete innerhalb der empfangenen Simulator-Transfer-Frames werden sukzessive interpretiert und ihre Werte entsprechend auf interne Variablen gesetzt. Beim Empfang des Kontrollframes `CF_LAST_ITEM` wird die Ausführung der On-board Kontrollalgorithmen für das definierte Zeitintervall der Lageregelungsfrequenz von 100 ms durch das Simulator-Frontend `SIM_FE` initiiert.
- Anschließend werden alle für die Satellitenkomponenten generierten Kommandos und Anfragen vom Simulator-Frontend `SIM_FE` in Simulator-Transfer-Frames verpackt und gefolgt vom Kontrollframe `CF_LAST_ITEM` zum Systemsimulator gesendet.

- Das FPGA-Entwicklungsboard Frontend-Modell `OBCdev_FE` des Simulators empfängt alle gesendeten Daten und leitet sie intern zu den entsprechenden Satellitenkomponentenmodellen weiter. Nach dem Empfang des Kontrollframes `CF_LAST_ITEM` beendet das FPGA-Entwicklungsboard Frontend-Modell `OBCdev_FE` seinen Empfangszustand und geht über zum nächsten Taktzyklus.

In dieser Token-basierten Ablaufsteuerung ist wichtig, dass alle Aufrufe von anderen Prozessen synchron erfolgen, d. h. der aufrufende Prozess wartet, bis der aufgerufene Prozess beendet wird. Erst dann läuft der aufrufende Prozess weiter. Dieses Prinzip ist im UML Sequenzdiagramm in Abbildung 3.9 durch die synchronen Aufrufpfeile mit gefüllten Spitzen verdeutlicht.

Synchronisierte Ablaufsteuerung parallel laufender Simulationskomponenten

Das beschriebene Prinzip der synchronisierten Ablaufsteuerung ist im Eigentlichen keine Ablaufsteuerung mehr, sondern stellt eine Synchronisation von unabhängigen Prozessen dar. Die beiden Prozesse werden im hier behandelten Anwendungsfall durch den Systemsimulator und das FPGA-Entwicklungsboard repräsentiert. Beide Elemente sind durch eine Schnittstelle zur Datenkommunikation miteinander verbunden, wobei es keine spezielle Flusskontrolle der Daten mehr gibt. Abbildung 3.10 stellt dieses Prinzip in einem UML Sequenzdiagramm dar. Das UML Diagramm stellt die asynchrone Datenkommunikation der beiden Prozesse durch asynchrone Aufrufpfeile mit nicht gefüllten Spitzen dar.

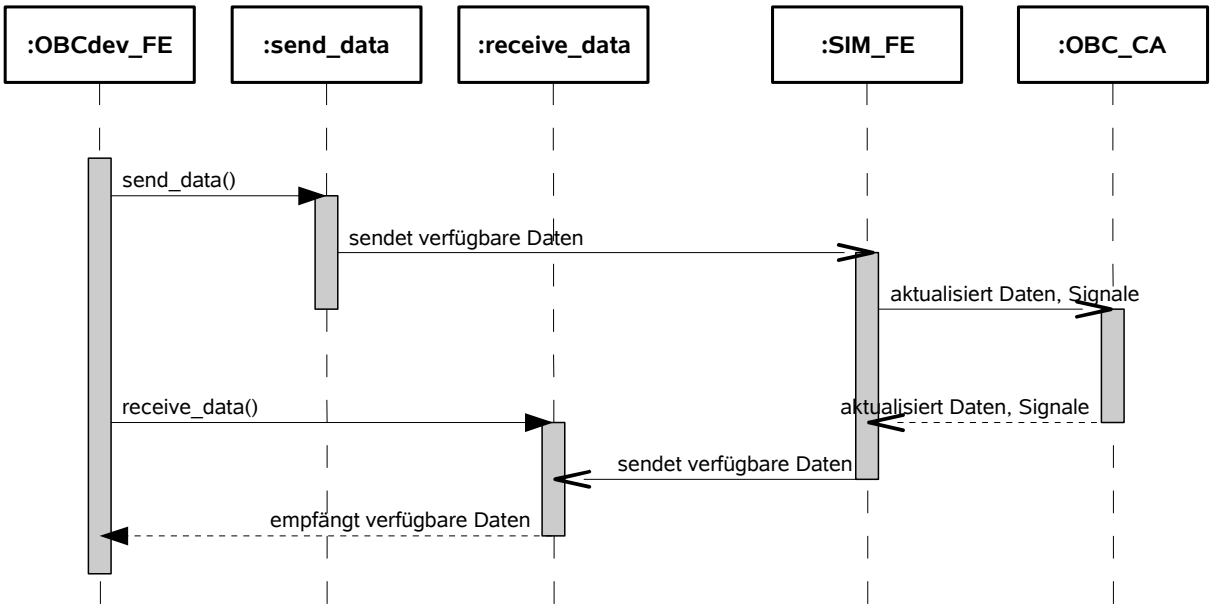


Abbildung 3.10: UML Sequenzdiagramm des parallelen Synchronisationsprinzips

Auf der Seite des Systemsimulators verpackt das FPGA-Entwicklungsboard Frontend-Modell `OBCdev_FE` während eines Taktzyklus alle verfügbaren Daten der Satellitenkomponentenmodelle in Simulator-Transfer-Frames und sendet sie zum FPGA-Entwicklungsboard. Anschließend werden direkt alle inzwischen empfangenen Daten aus den Simulator-Transfer-Frames ausgepackt, interpretiert und zu den entsprechenden Satellitenkomponentenmodellen weitergeleitet. Der Prozess der Datenkommunikation endet hier selbstständig und wird erst im nächsten Taktzyklus wiederholt.

Die Kontrollalgorithmen des FPGA-Entwicklungsboards bestehen aus vielen parallel laufenden Prozessen. Überwachungsfunktionen, Aufgaben der einzelnen Subsysteme und darunter auch die Lageregelungsalgorithmen werden zyklisch ausgeführt. Parallel dazu läuft das Simulator-Frontend `SIM_FE`, welches sich unabhängig von den anderen Prozessen um die Datenkommunikation mit dem Systemsimulator kümmert und die internen Variablen entsprechend der Kommunikation aktualisiert. Innerhalb einer Hauptanwendung werden die erwähnten Prozesse parallel zueinander gestartet. Die einzelnen Prozesse kommunizieren intern über Steuersignale und ähnlichen Mechanismen. Das Simulator-Frontend `SIM_FE` ist so implementiert, dass die restlichen On-board Kontrollalgorithmen nicht modifiziert werden müssen und den realen, für den späteren On-board Computer vorgesehenen Algorithmen entsprechen.

Empfängt das Simulator-Frontend `SIM_FE` einen Simulator-Transfer-Frame, so packt es diesen aus, interpretiert ihn und setzt die übertragene Information auf die entsprechende Systemvariable. Dabei werden ggf. entsprechende Steuersignale aktualisiert. Ändert zum Beispiel der Lageregelungsalgorithmus ein zuständiges Steuersignal, so reagiert das Simulator-Frontend `SIM_FE` umgehend, erzeugt das geforderte Datenpaket, verpackt es in einen Simulator-Transfer-Frame und sendet es über die Datenschnittstelle an den Systemsimulator.

In diesem Ablauf- bzw. Simulationssteuerungsprinzip sind zwei Dinge wichtig: Zum Einen muss die Synchronisation der beiden beteiligten Computersysteme, des Systemsimulators und des FPGA-Entwicklungsboards, gewährleistet sein. Zum Anderen muss aufgrund der Bündelung der Datenkommunikation durch eine Datenleitung sichergestellt werden, dass es dadurch zu keinen für die On-board Algorithmen relevanten Verzögerungen kommt. Für die ordnungsgemäße Ausführung der Lageregelungsalgorithmen heißt das zum Beispiel, dass der komplette benötigte Datenaustausch innerhalb der 100 ms dauernden Lageregelungsfrequenz erfolgen muss.

Die Synchronisation der beiden Computersysteme erfolgt über eine separate digitale Pulsverbindung, auf der vom FPGA-Entwicklungsboard ein konstantes Pulssignal erzeugt wird. Der Systemsimulator auf der anderen Seite empfängt das Pulssignal über eine digitale Schnittstellenkarte und synchronisiert sich selbstständig zu dem Signal. Daraus resultiert die Zuordnung des FPGA-Entwicklungsboards als Hauptsteuerungsorgan während der Systemsimulator jetzt die untergeordnete Rolle übernimmt. Spezielle Kontrollframes werden in diesem Zusammenhang dann nicht mehr benötigt.

3.3.1.3 Der Schnittstellenkarten-Treiber als separate Anwendung

Der Systemsimulator ist ganz bewusst so aufgebaut, dass die Satellitenkomponentenmodelle im Echtzeit-Simulator durch reine Abbildung in Software realisiert sind. Es erfolgt eine klare Trennung von den Softwaremodellen zu den Treibern der Hardwareschnittstellen. Dadurch wird sichergestellt, dass der Zugriff auf Computerhardware und evtl. damit verbundene Latenzzeiten unabhängig von der Echtzeit-Simulation der Satellitenkomponentenmodelle erfolgt. Demzufolge muss es wieder eine Kommunikationsschnittstelle zwischen dem Echtzeit-Simulator und dem Treiber der Hardwareschnittstelle geben, die über einen gemeinsamen Speicherbereich (Shared Memory) realisiert wird. Dieses Prinzip des modularen Aufbaus ermöglicht weiterhin ohne Änderungen im Quelltext den Aufbau von verschiedenen Testständ mit unterschiedlicher Hardware für die Datenübertragungsschnittstellen zum On-board Computer Simulator bzw. im späteren Projektverlauf dem realen On-board Computer.

Auch im hier behandelten Anwendungsbeispiel des Systemsimulators für den Kleinsatelliten *Flying Laptop* ist der Treiber für die RS422 Hardwareschnittstelle als separate Anwendung programmiert worden und wird parallel zum Echtzeit-Simulator gestartet. Der Treiber übernimmt alle Aufgaben bei der Kommunikation über die serielle Schnittstelle und kommuniziert mit dem Echtzeit-Simulator Frontend-Modell `OBCdev_FE` nur über einen gemeinsamen Speicherbereich. Die Klasse des gemeinsamen Speicherbereichs wird in beiden beteiligten Anwendungen gleichermaßen verwendet und ermöglicht eine bidirektionale und gleichzeitige Kommunikation.

Weiterhin übernimmt der RS422 Schnittstellenkartentreiber zusätzliche Sonderaufgaben. Zum Beispiel erfordern bestimmte Satellitenkomponenten durch ihren Schnittstellentyp eine sofortige Antwort auf übertragene Anfragen. In diesem Fall antwortet der Treiber umgehend mit bereits im Vorfeld zwischengespeicherten und vom Echtzeit-Simulator ständig aktualisierten Datenpaketen. Dies erfolgt selbstständig, ohne die entsprechende Anfrage an den Echtzeit-Simulator weiterzuleiten.

3.3.2 Modell des Sonnensensorsystems

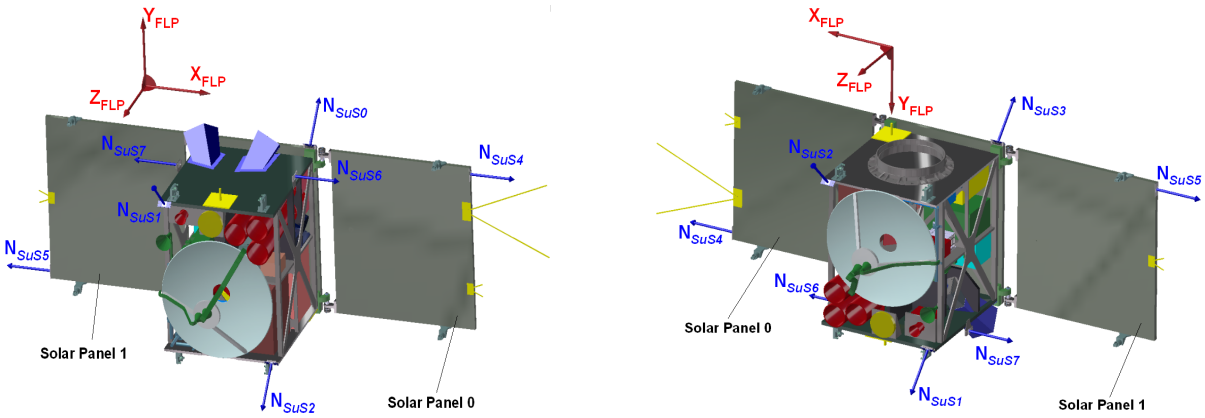


Abbildung 3.11: Position und Orientierung der Solarzellensensoren an der Satellitenstruktur

Das Sonnensensorsystem besteht aus zwei voneinander unabhängigen redundanten Elektroniken, die mit jeweils 8 Solarzellen verbunden sind. Die Solarzellen sind je 2 cm x 2 cm groß und stellen den Sensor dar, dessen erzeugter Strom über eine Spannungsmessung an einem Messwiderstand bekannter Größe bestimmt und in digitaler Form bereitgestellt wird. Je zwei Solarzellen sind an insgesamt 8 unterschiedlichen Positionen (siehe Abbildung 3.11) an der Satellitenstruktur montiert. Je Montageposition ist eine Solarzelle pro Ausleseelektronik vorhanden. Die Ausleseelektronik beinhaltet den angesprochenen Analog/ Digitalwandler und die Steuerungselektronik für den I^2C-Datenübertragungsbus, über den das System mit dem On-board Computer kommuniziert. Die eigentliche Intelligenz des Systems steckt im On-board Computer bzw. in den Kontrollalgorithmen, in denen aus den bereitgestellten Strömen der Solarzellensensoren der Vektor zur Sonne im körperfesten Koordinatensystem des Satelliten bestimmt wird. Die Genauigkeit des Systems liegt bei circa 5° und ist für die grobe Orientierung des Satelliten im `SAFE` Mode vorgesehen und ausreichend genau.

3.3.2.1 Prinzipielle Funktion des Modells

Aufgabe des Modells für das Sonnensensorsystem ist also in Abhängigkeit der aktuellen relativen Lage zur Sonne, die generierten Ströme an den einzelnen Solarzellensensoren im entsprechenden Format zur Abfrage durch den On-board Computer bereit zu stellen. Der generierte Strom an den Solarzellen ist abhängig von den folgenden Punkten:

- der Charakteristik der Solarzelle
- der Position in Relation zum Sonnenstand und zur Erde (aufgrund der Albedoreflektion) und der Entfernung zur Sonne
- der Temperatur der Solarzelle
- der Spannung der Solarzelle

Charakteristik der Solarzelle

Die Charakteristik der Solarzelle ist aus der Projektdokumentation entnommen worden und liegt im Modell in Form charakteristischer Parameter vor. Durch entsprechende Formeln, auf die später noch genauer eingegangen wird, ermittelt das Modell zur Laufzeit den gesuchten Wert aus den charakteristischen Parametern.

Lage und Position des Satelliten im Erdorbit

Den jeweils aktuellen Sonnenvektor, die Entfernung zur Sonne und den Vektor zur Erde im körperfesten Koordinatensystem stellt das Umgebungsmodell des Systemsimulators zur Verfügung.

Temperatur der Solarzelle

Die Temperatur der Solarzelle ist für die Bestimmung des generierten Stromes sehr wichtig, da dieser in hoher Abhängigkeit dazu steht und an den Solarzellensensoren im Verlauf einer Erdumkreisung große Temperaturunterschiede auftreten. Die Temperaturbestimmung der Solarzelle als Näherungslösung ist in mehrere Schritte aufgeteilt und beruht auf den folgenden Annahmen:

- Die Solarzellensensoren sind thermisch von der Satellitenstruktur entkoppelt.
- Die Wärmeabstrahlung des Satelliten selbst bleibt für die Temperaturbestimmung der Solarzellensensoren unberücksichtigt.
- Die Solarzellensensoren befinden sich zu jedem Zeitpunkt im Thermodynamischen Gleichgewicht.

Zuerst wird der Normalenvektor zur Solarzellensensoroberfläche aus einem Array mit allen Sensorpositionen entnommen. Anschließend kann der Winkel φ zwischen dem erwähnten Normalenvektor und dem Sonnenvektor, dem Winkel zwischen dem Normalenvektor und dem Erdvektor und der Winkel zwischen Sonnen- und Erdvektor berechnet werden. Die aktuelle Sonnenstrahlungsintensität $S_{vis,t}$ wird in Abhängigkeit der gesamten an der Sonne abgestrahlten Leistung und des Abstandes zur Sonne bestimmt. Bei diesem Schritt wird auch die mögliche Position des Satelliten im Erdschatten oder Erdhalbschatten berücksichtigt. Weiterhin wird der Konzentrationsfaktor c des Sonnenlichtes auf dem Solarzellensensor unter Berücksichtigung des Normalenvektors der Zelloberfläche und dem aktuellen Sonnenvektor

berechnet. Der Konzentrationsfaktor berücksichtigt weiterhin das Verhältnis zwischen der berechneten Intensität der Sonnenstrahlung $S_{vis,t}$ und der Intensität S_0 derselben unter Testbedingungen, für die die Charakteristiken der verwendeten Solarzellen gelten. Bei diesem Schritt fließt in die Berechnung auch ein, ob die Sonne vor oder hinter der Solarzelle steht.

$$c = max[0, \cos(\varphi)] \cdot \frac{S_{vis,t}}{S_0} \qquad \text{Formel 3.1}$$

c	Konzentrationsfaktor der Sonnenstrahlung auf der Solarzelle [-]
φ	Winkel zwischen Normalenvektor des Solarzellensensors und dem Vektor zur Sonne [°]
$S_{vis,t}$	Berechnete Sonnenstrahlungsintensität zum Zeitpunkt t [W/m²]
S_0	In der Solarzellencharakteristik angegebene Strahlungsintensität der Sonneneinstrahlung [W/m²]

Die Bestimmung der Solarzellensensortemperatur beruht auf dem Strahlungsgesetz nach Stefan Boltzmann bzw. der Umformulierung unter Berücksichtigung der Strahlungseigenschaften Grauer Körper, die eine Näherungslösung für reale Materialien darstellen.

$$\Phi_{Emission} = \epsilon_{IR} \cdot \sigma \cdot A \cdot T^4 \qquad \text{Formel 3.2}$$

$\Phi_{Emission}$	Vom Körper abgestrahlte Leistung [W]
ε_{IR}	Emissionskoeffizient im infraroten Spektralbereich [-]
σ	Stefan Boltzmann Konstante $\sigma = 5{,}67051 \cdot 10^{-8}\ W/m^2K^4$
A	Fläche des strahlenden Körpers [m²]
T	Temperatur des abstrahlenden Körpers [K]

Für das Verhältnis zwischen dem Absorptionskoeffizienten α und dem Emissionskoeffizienten ε gilt in der Raumfahrt nach GILMORE *„Spacecraft Thermal Control Handbook“* [Gilmore '02] für Graue Körper näherungsweise

$$\alpha_{vis}/\epsilon_{IR} \equiv 1{,}3 \quad bzw. \quad \alpha_{IR}/\epsilon_{IR} \equiv 1{,}0 \qquad \text{Formel 3.3}$$

α_{vis}	Absorptionskoeffizient im visuellen Spektralbereich [-]
α_{IR}	Absorptionskoeffizient im infraroten Spektralbereich [-]

Die abgestrahlte Leistung $\Phi_{Emission}$ der Solarzellensensoren wird jetzt wie folgt formuliert

$$\Phi_{Emission} = \epsilon_{IR} \cdot \sigma \cdot A_{Solarzelle} \cdot T^4_{Solarzelle} \qquad \text{Formel 3.4}$$

$A_{Solarzelle}$	Oberfläche des Solarzellensensors [m²]
$T_{Solarzelle}$	Temperatur des Solarzellensensors [K]

und für die gesamte absorbierte Leistung $\Phi_{Absorption}$ der Solarzellensensorfläche gilt bei Berücksichtigung der direkt von der Sonne absorbierten Strahlung, der an der Erdatmosphäre reflektierten Albedostrahlung und der Erdinfrarotstrahlung der nachfolgende Zusammenhang.

$$\Phi_{Absorption} = \alpha_{vis} \cdot A_{Solarzelle} \cdot S_{vis,t} \cdot c + d \cdot A_{Solarzelle} \cdot (S_{IR} \cdot \alpha_{IR} + S_{vis,t} \cdot af \cdot \alpha_{vis}) \qquad \text{Formel 3.5}$$

$\Phi_{Absorption}$	Vom Körper absorbierte Leistung [W]
d	Faktor zur Berücksichtigung der Sichtbarkeit der Erde von der aktuellen Solarzellensensorposition [-]
S_{IR}	Strahlungsintensität der Erdinfrarotstrahlung [W/m^2]
af	Albedofaktor [-]

Die Strahlungsintensität der Erdinfrarotstrahlung S_{IR} ist abhängig von der Orbithöhe des Satelliten und variiert in den angestrebten Referenzorbits zwischen 170 W/m^2 und 190 W/m^2. Zur Vereinfachung der Berechnung wird hier eine konstante Strahlungsintensität der Erdinfrarotstrahlung S_{IR} von 180 W/m^2 angenommen.

Der Albedofaktor *af* beschreibt den Anteil der Sonnenstrahlung, der von der Erdatmosphäre zurück in den Weltraum reflektiert wird. Nach Larson & Wertz *„Space Mission Analysis and Design“* [Larson & Wertz '99] kann man diesen im Mittel mit dem Faktor 0,3 beschreiben, wenn die gesamte sichtbare Erdhalbkugel von der Sonne beleuchtet wird. Aufgrund der verhältnismäßig geringen Orbithöhe des Satelliten relativ zum Erddurchmesser bleibt ein Sichtfaktor hier unberücksichtigt bzw. die Sichtbarkeitsbedingungen der Erdoberfläche von der aktuellen Solarzellensensorposition sind bereits im Faktor *d* berücksichtigt. Dadurch wird der Anteil, der aus der Perspektive des Satelliten bzw. des betroffenen Solarzellensensors sichtbaren bestrahlten Erdatmosphäre im eingeführten Albedofaktor berücksichtigt, denn nur der sonnenbeleuchtete Anteil kann auch Albedostrahlung verursachen. Die folgende Abbildung 3.12 verdeutlicht in einer groben Skizze die geometrischen Verhältnisse.

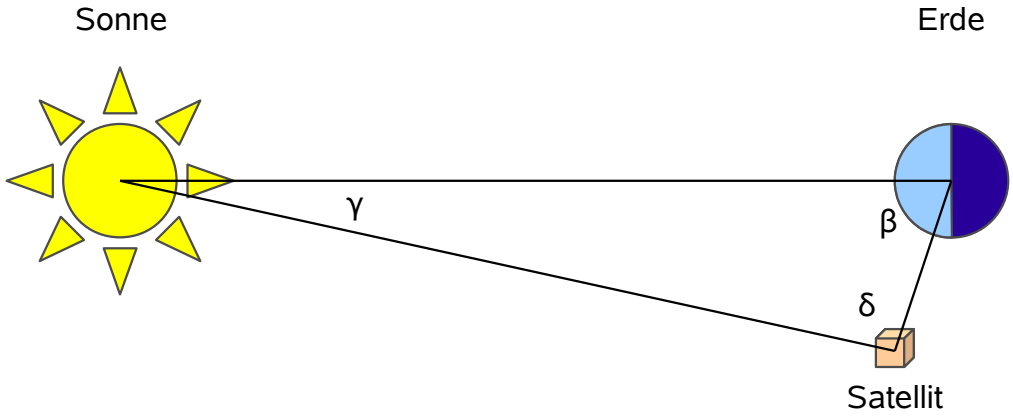


Abbildung 3.12: Nicht maßstäbliche Skizze der geometrischen Verhältnisse der Albedo Reflektion

Der beleuchtete Teil der aus Perspektive des Satelliten sichtbaren Erdoberfläche verhält sich nicht linear mit dem Blickwinkel β, kann aber über diesen linear angenähert werden. Wir betrachten also den Winkel β. Ist β gleich 0°, so ist die gesamte vom Satelliten sichtbare Erdoberfläche beleuchtet, wenn β gleich 90° ist, dann sind nur 50% beleuchtet und wenn β

gleich 180° ist, dann sind 0% der vom Satelliten sichtbaren Erdoberfläche beleuchtet. Weiterhin können wir annehmen, dass aufgrund des sehr großen Abstandes zwischen Sonne und Erde und des vergleichsweise geringen Abstandes zwischen Erde und Satellit der Winkel γ nahezu Null ist ($\gamma \approx 0°$). Daher ist die Summe aus β und γ näherungsweise gleich 180°, da 180° die Summe aller Winkel innerhalb eines Dreiecks ist. Jetzt kann man den Albedofaktor *af* wie folgt formulieren.

$$af = 0{,}3 \cdot \frac{(180° - \beta)}{180°} = 0{,}3 \cdot \frac{\delta}{\pi} \qquad \text{Formel 3.6}$$

Durch die Annahme des Thermodynamischen Gleichgewichts können wir Formel 3.4 und Formel 3.5 gleichsetzen und nach der Temperatur *T* auflösen.

$$T_{Solarzelle} = \sqrt[4]{\frac{1{,}3 \cdot S_{vis,t} \cdot c + d \cdot (180 + 1{,}3 \cdot S_{vis,t} \cdot 0{,}3 \cdot \delta / \pi) + 15 \frac{W}{m^2}}{\sigma}} \qquad \text{Formel 3.7}$$

Die berücksichtigte absorbierte Wärmeleistung von 15 W/m^2 des Solarzellensensors beinhaltet eine Restwärmeleitung von der Satellitenstruktur und den Strahlungsaustausch mit der Weltraumhintergrundstrahlung. Dieser Term führt auch dazu, dass die Solarzellensensoren nicht unter -150° C abkühlen.

Spannung der Solarzelle

Der verbleibende unbekannte Parameter ist die Zellspannung der Solarzelle unter der Sonneneinstrahlung und einer bekannten Orientierung zur Sonneneinstrahlrichtung. Aus der Zellcharakteristik ist die maximale Leistung P_{mp}, die Spannung und der Strom der Solarzelle bei einer definierten Sonnenstrahlungsintensität bekannt. Wenn man jetzt die Sonnenstrahlungsintensität bei bekannter Solarzellenleistung und die aktuell berechnete Strahlungsintensität über den Konzentrationsfaktor *c* untereinander ins Verhältnis setzt, erhält man die aktuelle Solarzellenleistung $P_{Solarzelle}$.

$$P_{mp} \cdot (c + d \cdot af) = P_{Solarzelle} \qquad \text{Formel 3.8}$$

Mit dem Ohmschen Gesetz

$$R = \frac{U}{I} \qquad \text{Formel 3.9}$$

R	Elektrischer Widerstand [Ω]
U	Spannung [V]
I	Strom [A]

und der Formel für die Elektrische Leistung *P*

$$P = U \cdot I \qquad \text{Formel 3.10}$$

kann jetzt eine Beziehung zwischen der Solarzellenleistung $P_{Solarzelle}$ und der Spannung $U_{Solarzelle}$ wie folgt formuliert

$$P_{mp} \cdot (c + af \cdot d) = P_{Solarzelle} = \frac{U^2_{Solarzelle}}{R_{Messwiderstand}} \qquad \text{Formel 3.11}$$

und nach der Solarzellenspannung aufgelöst werden.

$$U_{Solarzelle} = \sqrt{P_{mp} \cdot (c + af \cdot d) \cdot R_{Messwiderstand}} \qquad \text{Formel 3.12}$$

Der Messwiderstand $R_{Messwiderstand}$ an der Solarzelle ist ein bekannter Parameter des Sonnensensorsystems. Somit ist die aktuelle Spannung der Solarzellensensoren bei den gegebenen Sonneneinstrahlbedingungen bestimmt.

Der generierte Strom des Solarzellensensors

Bei der Berechnung des gesamten generierten Stromes am Solarzellensensor liefert LaRoche *„Solargeneratoren für die Raumfahrt“* [LaRoche '97] wertvolle Gleichungen. Mit Hilfe seiner Gleichung (3.13) kann der gesuchte Strom basierend auf bereits bekannten oder leicht ermittelbaren Größen bestimmt werden.

$$I_{Solarzelle} = I_{sc} - I_0 \cdot \left(e^{\frac{U_{Solarzelle}}{U^*_T}} - 1\right) \qquad \text{Formel 3.13}$$

I_{sc}	Kurzschlussstrom [A]
I_0	Sättigungsstrom [A]
U^*_T	Tatsächliche thermische Spannung [V]

Während der Parameter I_{sc} leicht messbar und dementsprechend auch Teil der gegebenen Solarzellencharakteristik ist, gelten für die tatsächliche thermische Spannung U^*_T und den Sättigungsstrom I_0 wieder nach LaRoche die folgenden Beziehungen.

$$U^*_T = U_{mp} \cdot \frac{I_{sc} - I_{mp}}{I_{mp}} \qquad \text{Formel 3.14}$$

U_{mp}	Spannung am maximalen Leistungspunkt [V]
I_{mp}	Strom am maximalen Leistungspunkt [A]

$$I_0 = \frac{I_{sc}}{e^{\frac{U_{oc}}{U^*_T}} - 1} \qquad \text{Formel 3.15}$$

U_{oc}	Kurzschlussspannung [V]

Die hier neu eingeführten Parameter U_{mp}, I_{mp}, und U_{oc} sind wieder leicht messbar und dementsprechend ebenfalls Teil der gegebenen Solarzellencharakteristik. Somit ließe sich der Strom der Solarzellensensoren bei den gegebenen Sonneneinstrahlbedingungen bestimmen. Wichtig ist hier jedoch die Berücksichtigung der Temperaturabhängigkeit der charakteristischen Parameter U^*_T, I_{sc} und I_0, deren Werte eigentlich nur für eine bestimmte Normtemperatur T_0 in der Solarzellencharakteristik bekannt sind. Bevor man sie berechnen kann, müssen erst die temperaturabhängigen Ausgangsparameter in Abhängigkeit von der aktuellen Temperatur T zum Messungszeitpunkt bestimmt werden. Dies wird unter Verwendung von Formel 3.14 und Formel 3.15 mit den nachfolgenden Gleichungen durchgeführt und die erhaltenen Parameter dann zur weiteren Berechnung verwendet.

$$I_{sc}(T) = I_{sc}(T_0) + \frac{dI_{sc}}{dT} \cdot (T - T_0) \qquad \text{Formel 3.16}$$

$$I_{mp}(T) = I_{mp}(T_0) + \frac{dI_{mp}}{dT} \cdot (T - T_0) \qquad \text{Formel 3.17}$$

$$U_{mp}(T) = U_{mp}(T_0) + \frac{dU_{mp}}{dT} \cdot (T - T_0) \qquad \text{Formel 3.18}$$

$$U_{oc}(T) = U_{oc}(T_0) + \frac{dU_{oc}}{dT} \cdot (T - T_0) \qquad \text{Formel 3.19}$$

Die in den obigen Gleichungen noch unbekannten Temperaturkoeffizienten sind Teil der vom Hersteller bereitgestellten Solarzellencharakteristik.

Schließlich wird auf den berechneten Strom $I_{Solarzelle}$ noch ein Signalrauschen künstlich aufgebracht, um den realen Messbedingungen zu entsprechen. Das künstliche Signalrauschen wird als Zufallswert innerhalb einer maximalen Differenz bestimmt und ist so gewählt, dass eine bestimmte maximale Abweichung vom maximalen Strom der Solarzellensensoren nicht überschritten wird.

3.3.2.2 Kommunikation

Zur Verdeutlichung der Kommunikation mit dem On-board Computer wird hier nochmals näher auf die Ausleseelektronik des Sonnensensorsystems eingegangen. Wie bereits oben erwähnt, besteht sie aus zwei baugleichen redundanten Einheiten, die mit den 16 Solarzellensensoren verbunden sind. Je einer der Analog/ Digitalwandler (ADC) ist jeweils mit 8 Solarzellensensoren an 8 verschiedenen Montagepositionen verbunden, ein zweiter mit den verbleibenden 8 Solarzellensensoren. Die Analog/ Digitalwandler werden von je einem I^2C-Buscontrollern (PCF) ausgelesen, die an einem I^2C-Bus zum On-board Computer angeschlossen sind. Dabei sind die beiden Buscontroller mit unterschiedlichen Busadressen konfiguriert, so dass sie unabhängig voneinander vom On-board Computer abgefragt werden können. Die Kommunikation geht immer vom On-board Computer aus, der nacheinander die beiden I^2C-Buscontroller abfragt. Die nachfolgende Abbildung 3.13 verdeutlicht die logische Struktur und die Kommunikationsverbindungen des Sonnensensorsystems.

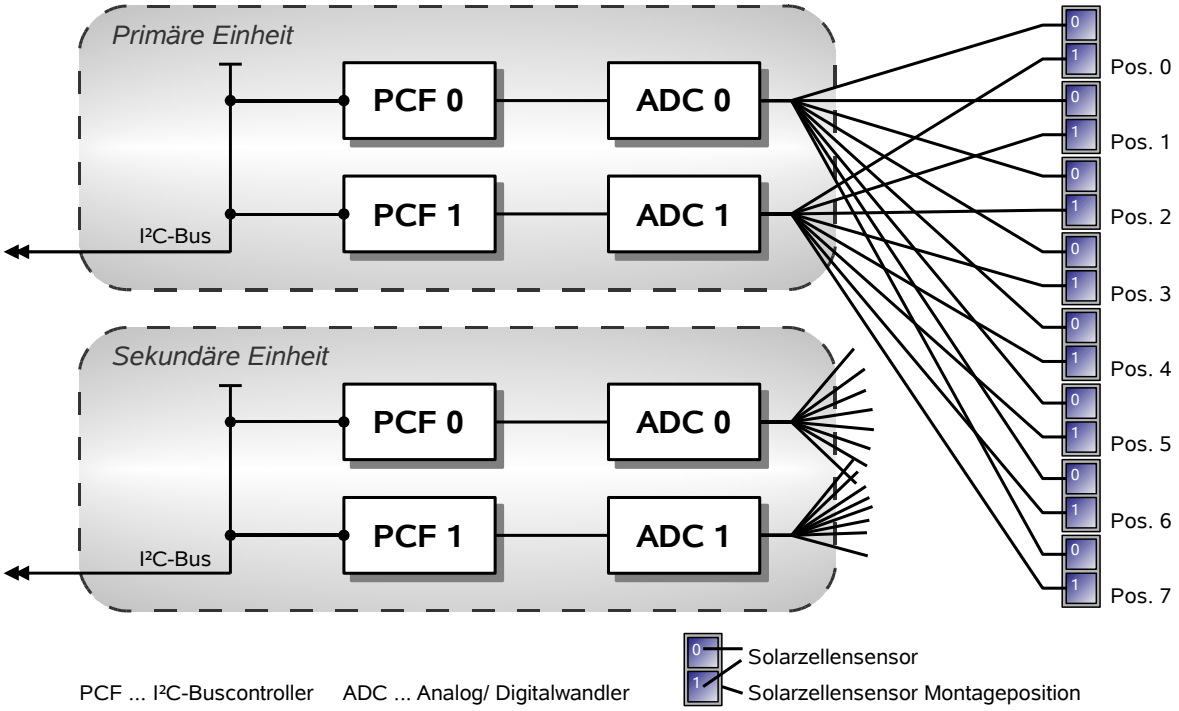


Abbildung 3.13: Struktur und Kommunikationsverbindungen des Sonnensensorsystems

Die beiden redundanten Teile des Sonnensensorsystems sind im Systemsimulator durch zwei unabhängige Instanzen realisiert. Sie liefern dem On-board Computer auf Anfrage die aktuellen Sensorströme, woraufhin die On-board Algorithmen entscheiden, welche Ströme sie für die Bestimmung des Sonnenvektors verwenden werden. Die Analog/ Digitalwandler wandeln die Ströme in 12-bit breite digitale Werte um, jedoch werden die Ströme von den I²C-Buscontrollern als 16-bit Werte übertragen. Die dabei hinzugefügten Bits werden wertigkeitsneutral mit Nullen aufgefüllt. Tabelle 3.9 unten zeigt die Telemetrie Format Definition, wie sie vom Sonnensensorsystem auf Anfrage übertragen wird. Das Sonnensensorsystem stellt diese aktuelle Telemetrieinformation im Systemsimulator intern zur Weiterübertragung durch ein entsprechendes I²C-Bus Frontend an den On-board Computer zur Verfügung.

Tabelle 3.9: Telemetrie Format des Sonnensensorsystems

Gesamte Telemetrie																		
Erstes TM Paket (von PCF 0)									Zweites TM Paket (von PCF 1)									
Ziel Adresse von PCF 0	Strom an Pos. 0 (Zelle 0)	Strom an Pos. 1 (Zelle 0)	Strom an Pos. 2 (Zelle 0)	Strom an Pos. 3 (Zelle 0)	Strom an Pos. 4 (Zelle 0)	Strom an Pos. 5 (Zelle 0)	Strom an Pos. 6 (Zelle 0)	Strom an Pos. 7 (Zelle 0)	Ziel Adresse von PCF 1	Strom an Pos. 0 (Zelle 1)	Strom an Pos. 1 (Zelle 1)	Strom an Pos. 2 (Zelle 1)	Strom an Pos. 3 (Zelle 1)	Strom an Pos. 4 (Zelle 1)	Strom an Pos. 5 (Zelle 1)	Strom an Pos. 6 (Zelle 1)	Strom an Pos. 7 (Zelle 1)
1 Byte	2 Byte	2	2	2	2	2	2	2	1 Byte	2 Byte	2	2	2	2	2	2	2

3.4 Aufbau des zentralen Missionskontrollsystems

Der einzige Weg einen Satelliten im Erdorbit zu kontrollieren, ist die Kontrolle über eine Schnittstelle zur Datenübertragung. Bisher ist dies als Funkverbindung (alternativ zu einer laser-optischen Datenübertragung) realisiert, über die Kontroll- und Zustandssignale übertragen werden. Auf der Erdoberfläche wird zu diesem Zweck ein Missionskontrollsystem (MCS) verwendet. Es bildet die Schnittstelle zwischen den Satellitenbedienern und dem Satelliten selbst. Zur Datenübertragung mit dem Satelliten ist ferner die Bodenstation involviert, die aus der notwendigen Antennenanlage auf dem Boden besteht. Letztere muss sich nicht am gleichen Ort, sondern kann sich auch räumlich entfernt befinden. Sie ist dann lediglich durch eine sichere, meist bodengebundene Datenübertragungsleitung mit dem Kontrollsystem verbunden.

Zur Bedienung des Satelliten muss das Missionskontrollsystem eine Vielzahl von verschiedenen Aufgaben erfüllen und den Benutzer in der Steuerung des und der Kommunikation mit dem Satelliten zu unterstützen. Zum Beispiel muss es aktuelle Zustandsinformationen des Satelliten und der Nutzlast zur Überprüfung durch den Operator bereithalten. Der Satellit sendet daher in regelmäßigen Abständen die wichtigsten Zustandsinformationen zum Kontrollsystem. Diese werden dort vor-prozessiert und dem Operator in übersichtlicher Form visualisiert.

Weiterhin muss das Kontrollsystem in der Lage sein, große Mengen an wissenschaftlichen Daten zu empfangen und für die weitere Prozessierung zu speichern. Eine große Datenbank ist für diese Aufgabe verantwortlich und speichert sämtliche vom Satelliten empfangenen Telemetrie-Daten. Auf der Basis dieser Telemetrie können detaillierte Missionsanalysen im Hinblick auf die Systemperformance durchgeführt werden. Aber auch in Fehlerfällen ist die Telemetrie sehr hilfreich bei der Identifikation der Fehlfunktion und ihrer Ursache.

Die Missions-Datenbank enthält weiterhin alle erlaubten Kommandos, die an den Satelliten gesendet werden können. Somit ist es dem Satellitenoperator möglich, sich aus diesem Kommandosatz geeignete Kommandos zu einer Sequenz zusammenzustellen, um den Satelliten die gewünschte Aufgabe durchführen zu lassen. Das Missionskontrollsystem kann aber auch direkt mit einem Satellitensimulator verbunden werden, um Kommandos bzw. Kommandosequenzen zu testen bevor sie zum echten Satelliten gesendet werden. Durch dieses Verfahren wird gewährleistet, dass keine Kommandos zum Satelliten gesendet werden, die diesen gefährden könnten.

3.4.1 Selektion eines geeigneten Systems

In Europa existieren mehrere kommerzielle bzw. nicht-kommerzielle Missionskontrollsysteme: zum Beispiel das „Spacecraft Control and Operation System 2000“ von der ESA (besser bekannt als SCOS-2000), das „Central Checkout System“ und „OpenCenter“. Für die Verwendung mit dem Systemsimulator MDVE am Institut für Raumfahrtsysteme in Stuttgart haben wir uns für die Verwendung von SCOS-2000 entschieden. SCOS-2000 wird unter Federführung der ESA entwickelt und ist mittels einer kostenlosen Lizenz verfügbar. Es wird von der ESA unter anderem bei den Missionen Integral, Mars Express und SMART-1 verwendet. In Abbildung 3.14 unten sieht man ein ausgeführtes SCOS-2000 System mit ein paar Beispielen von alphanumerischen oder graphischen Telemetrie-Visualisierungsfenstern.

Alphanumerische Anzeigen stellen ausgewählte Parameter mit ihrem Namen und dem jeweils aktuellsten Wert dar. Gleichzeitig sind Informationen über die Aktualität des Parameterwertes verfügbar, d. h. in welchem Telemetriepaket und zu welchem Zeitpunkt er zuletzt empfangen wurde. Die graphischen Anzeigen stellen den Verlauf eines oder mehrerer Parameter über einen konfigurierbaren Zeitraum dar. Werden neue Parameterwerte empfangen, so werden diese sofort im graphischen Verlauf eingezeichnet. Informationen über die Aktualität des jüngsten Parameterwertes sind ebenfalls verfügbar. Sowohl die alphanumerischen als auch die graphischen Anzeigen können je nach Belieben des Operators dynamisch während der Systemausführung konfiguriert werden.

Abbildung 3.14: SCOS-2000 mit ein paar Beispielen von Telemetrie Visualisierungsfenstern

SCOS-2000 ist ein klassisches verteiltes System, welches auf einer Vielzahl von durch das Netzwerk verbundenen Servern und Einzel-PC läuft. Allerdings kann es auch auf einem einzelnen PC installiert und ausgeführt werden. Das System kann während der Laufzeit dynamisch konfiguriert werden, was insbesondere eine höhere Flexibilität während der Mission ermöglicht. Aus Sicherheitsgründen bietet es eine eigenständige Benutzerverwaltung.

Die systemeigene Datenbank ist für die Aufzeichnung von Daten über einen langen Missionszeitraum ausgelegt und kann von anderen SCOS-2000 Unterprogrammen oder über eine externe Schnittstelle ausgelesen und ausgewertet werden. Es unterstützt standardmäßig sowohl den bei der ESA verbreiteten CCSDS Telemetrie und Telekommando Standard als

auch den Packet Utilization Standard (PUS). Qualität und Vollständigkeit aller empfangenen Telemetrie Pakete werden automatisch überprüft und die Korrektheit der zu sendenden Telekommandos wird direkt beim Laden der Kommandos verifiziert.

3.4.2 Installation und Konfiguration

Von der ESA wird SCOS-2000 im Quelltext ausgeliefert, um eine einfache Anpassung an eigene Bedürfnisse des Systems zu ermöglichen. Der erste Schritt bei der Installation ist demzufolge die Kompilation des Quelltexts in ausführbare Programme. Dieser Schritt war recht kompliziert und nicht intuitiv, da in den ersten Versionen einige Fehler und Unzulänglichkeiten zwischen den ESA Installationshinweisen und dem Quelltext bestanden. Da am Institut für Raumfahrtsysteme SuSE Linux 9.3 verwendet wird, ließen sich die von der ESA für SuSE Linux Enterprise Server 9 bereitgestellten Installationskripte leider auch nicht direkt adaptieren.

Die schließlich erfolgreiche Installation des Missionskontrollsystems während der Diplomarbeit von Alexander Brandt [Brandt '07] wurde als Einzel-PC Installation konfiguriert. Weitere Konfigurationsobjekte sind die Netzwerkeinstellungen und die Mission Information Base (MIB), welche die missionsspezifischen Telekommando- und Telemetriedefinitionen des Projekts enthält. Die Konfiguration erfolgt über spezielle Konfigurationsdateien. Diese Dateien werden am IRS automatisch aus einer Datenbank erzeugt und können anschließend direkt in SCOS-2000 importiert werden. Beim Importieren wird die Konsistenz der Konfigurationsdateien überprüft und Fehlkonfigurationen sofort angezeigt.

Zur Erleichterung von weiteren Installationen bzw. der Installation von aktuellen Programmversionen ist eine schrittweise Anleitung für die Projektdokumentation erstellt worden. Sie beinhaltet in ausführlicher Form jeden notwendigen Installationsschritt, begonnen von der Kompilation und Installation der zusätzlichen Softwarepakete bis hin zur Konfiguration der SCOS-2000 Netzwerkeinstellungen.

3.4.3 Verbindung des Missionskontrollsystems mit dem Systemsimulator

Das Missionskontrollsystem SCOS-2000 soll in der MDVE Systemsimulator Umgebung die Aufgaben des Zentralen Kontrollsystems (CCS) übernehmen. Typischerweise ist SCOS-2000 als Missionskontrollsystem nicht direkt mit der Antennenanlage der Bodenstation verbunden. Diese Datenverbindung wird bei der ESA aufgrund der Vielzahl von Antennenanlagen über ein so genanntes „Netzwerk Kontroll- und Telemetrie Leitungssystem“ (Network Control and Telemetry Routing System, NCTRS) hergestellt. Das NCTRS bietet standardisierte Schnittstellen zu den Antennenanlagen und SCOS-2000 und leitet die Daten in transparenter Art und Weise von der Quelle zu ihrem dedizierten Ziel. Von der ESA wird das NCTRS ebenfalls unter einer kostenlosen Lizenz angeboten, ist aber nur für Sun/Solaris Betriebssysteme verfügbar.

Im MDVE Labor am Institut für Raumfahrtsysteme ist für die erwähnte Datenverbindung ein anderer Weg gewählt worden, um den erhöhten Aufwand für die Implementierung des NCTRS zu vereinfachen. Alle in diesem Kontext relevanten Funktionen des NCTRS wurden in einem eigenständigem Leitungssystem, dem so genannten „SCOS Proxy“ (siehe Abbildung 3.15) umgesetzt. Es bietet ebenfalls standardisierte und transparente Schnittstellen über TCP/IP Verbindungen zwischen den Kommunikationsteilnehmern. SCOS-2000 ist über die bereits erwähnten Konfigurationsdateien so konfiguriert, dass es sich statt mit dem NCTRS

mit dem SCOS Proxy verbindet. Auf der anderen Seite ist sowohl der Systemsimulator als auch die Telemetrie- und Telekommando-Schnittstelle des simulierten Satelliten mit dem SCOS Proxy verbunden.

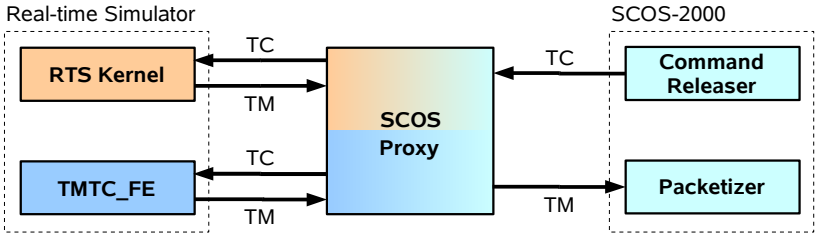


Abbildung 3.15: Verbindungselemente zwischen dem Missionskontrollsystem SCOS-2000 und dem MDVE Systemsimulator

Aufgabe des SCOS Proxy ist es nun, alle gesendeten Pakete zu ihrem Ziel zu leiten und dabei ggf. Änderungen in den Paket Kopfdaten bzw. Umpaketierungen durchzuführen. Die Unterscheidung des Telekommandoziels erfolgt über die „Application Process Identification“ (APID), die im PUS Paketkopf enthalten ist. Kommandos an den Systemsimulator beispielsweise werden von SCOS-2000 direkt im PUS Paket Standard erzeugt und vom SCOS Proxy unverändert an den Systemsimulator weitergeleitet. Demgegenüber werden Telekommandos an den simulierten Satelliten aus dem PUS Paket ausgepackt und nur im eigenständigen Paketformat des Projekts weitergeleitet, da der SCOS Proxy die Funkstrecke zwischen der Antennenanlage und dem Satelliten umgeht. Erreicht ein Telemetriepaket des simulierten Satelliten den SCOS Proxy, so wird es dort erst in ein PUS Paket und anschließend ins CCSDS Paketformat eingepackt und erst dann an SCOS-2000 weitergeleitet. Der Programmbaustein zum Empfang von Datenpaketen in SCOS-2000 ist der sogenannte „Packetizer“.

3.4.4 Bedienung von simuliertem Satellit und Systemsimulator

Als zentrales Kontrollsystem wird das Missionskontrollsystem SCOS-2000 sowohl für die Kontrolle des simulierten Satelliten als auch für die Steuerung des Systemsimulators verwendet. Die Kommandos werden in beliebiger Reihenfolge in derselben Oberfläche ausgewählt, ggf. modifiziert und abgesendet. Dazu sind in der Mission Information Base, der missionsspezifischen Telemetrie und Telekommando Datenbank, die entsprechenden Kommandos definiert und dem Operator somit zur Auswahl verfügbar.

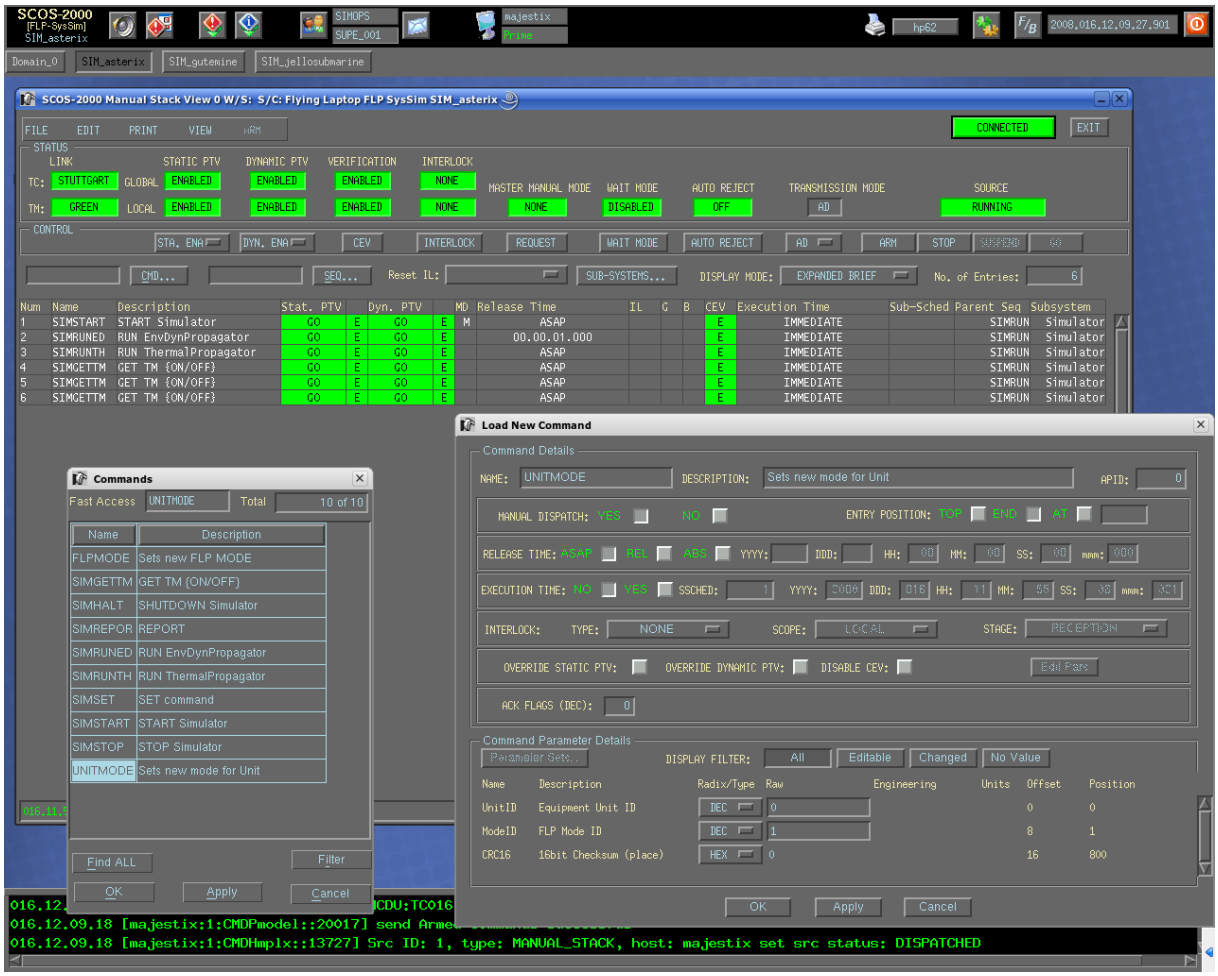
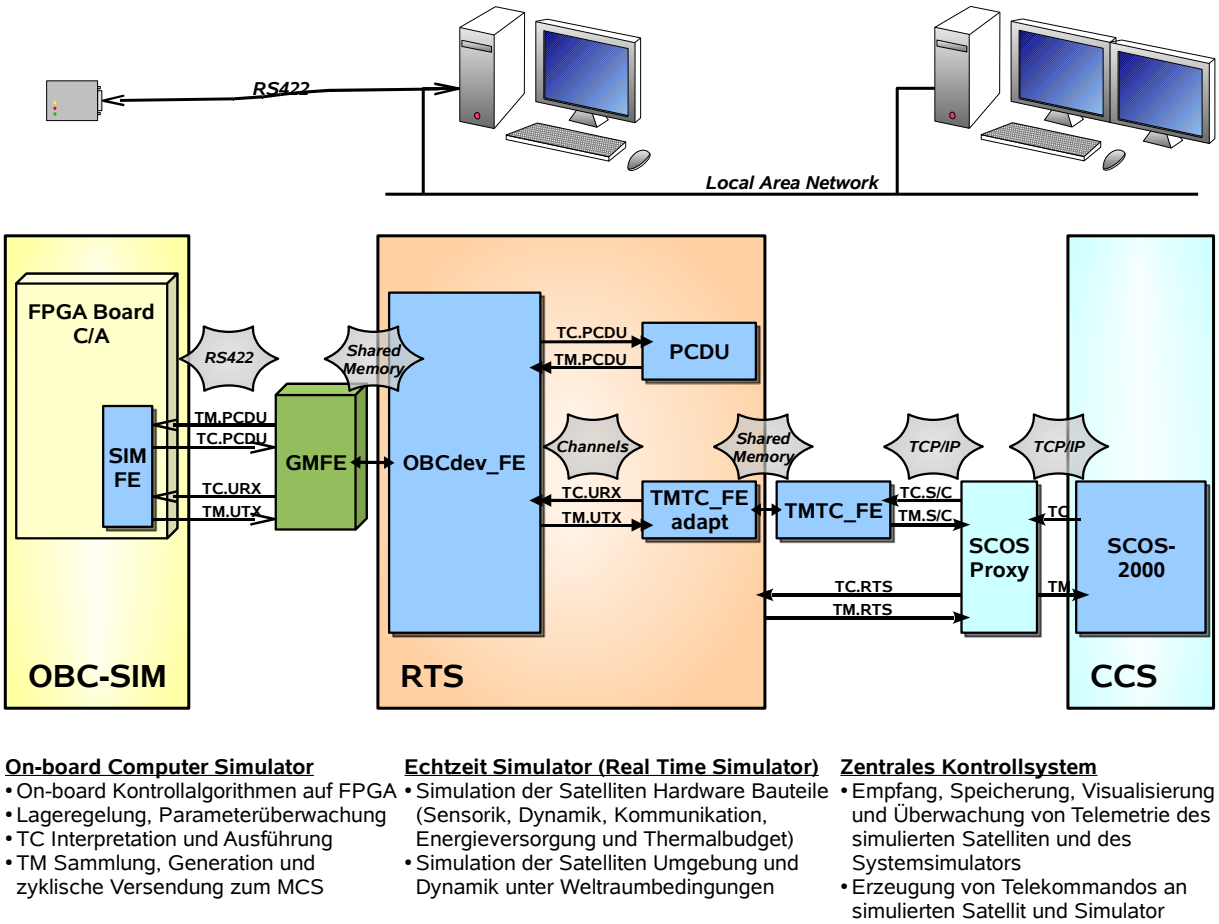


Abbildung 3.16: SCOS-2000 mit Fenstern zur Auswahl und Absendung von Telekommandos

Die jeweiligen graphischen Menüs für die Auswahl eines Kommandos, das eventuelle Editieren von einzelnen Parametern im ausgewählten Telekommando und die Liste aller erstellten aber noch nicht versendeten Kommandos sind aus Abbildung 3.16 ersichtlich. Im Fenster „Commands“ stehen eine Reihe von definierten Telekommandos zur Verfügung, die nach Auswahl im Fenster „Load New Command“ ggf. editiert werden können bevor sie in den „SCOS-2000 Manual Stack View“ geladen werden. Dort stehen die Telekommandos dann zur manuellen Ausführung bereit, d. h. sie werden vom Operator von Hand aktiviert (Button „Arm“ im Bereich rechts oben) und anschließend abgesendet (Button „Go“). Beim Aktivieren des Kommandos wird es auf inhaltliche Konsistenz und ggf. auf weitere vordefinierte Bedingungen überprüft. Erst wenn alle Bedingungen erfüllt und alle Felder grün aufleuchten, kann das Kommando abgesendet werden. Im Beispiel in Abbildung 3.16 ist eine Kommandosequenz von sechs einzelnen Telekommandos an den Systemsimulator geladen. Durch die Abfolge wird der Simulator, der Umgebungs- und Thermalpropagator gestartet und anschließend die zyklische Sendung von drei unterschiedlichen Telemetriepaketen aktiviert.

Die gleichzeitige Bedienung von simuliertem Satellit und Systemsimulator von einer Oberfläche aus ermöglicht die Generierung und Durchführung von Testfällen in automatisierten Prozeduren. Somit können bei Änderungen in der Satelliten On-board Kontrollalgorithmen die Verifikations-Tests mit geringem Aufwand wiederholt werden.



On-board Computer Simulator
- On-board Kontrollalgorithmen auf FPGA
- Lageregelung, Parameterüberwachung
- TC Interpretation und Ausführung
- TM Sammlung, Generation und zyklische Versendung zum MCS

Echtzeit Simulator (Real Time Simulator)
- Simulation der Satelliten Hardware Bauteile (Sensorik, Dynamik, Kommunikation, Energieversorgung und Thermalbudget)
- Simulation der Satelliten Umgebung und Dynamik unter Weltraumbedingungen

Zentrales Kontrollsystem
- Empfang, Speicherung, Visualisierung und Überwachung von Telemetrie des simulierten Satelliten und des Systemsimulators
- Erzeugung von Telekommandos an simulierten Satellit und Simulator

Abbildung 3.17: Datenflüsse und Verbindungsarten zwischen den beteiligten Elementen von Missionskontrollsystem, Echtzeit-Simulator und On-board Computer Simulator

Die Datenflüsse und Verbindungsarten zwischen den beteiligten Elementen von Missionskontrollsystem und Simulator sind in Abbildung 3.17 am Beispiel der Energieversorgungseinheit dargestellt. In kurzen Worten ist weiterhin die primäre Aufgabe der jeweiligen Teile erläutert. Die Energieversorgungseinheit (PCDU) zum Beispiel sendet in regelmäßigen Abständen ihren aktuellen Status zum On-board Computer. Dieser speichert die Statusinformation der PCDU und sendet sie wiederum in regelmäßigen Abständen zum Missionskontrollsystem. Im Systemsimulator sieht der dabei zurückgelegte Weg der Daten wie folgt aus: Das Modell der PCDU im Echtzeit-Simulator erzeugt turnusmäßig die entsprechende Telemetrie und sendet sie über einen internen Kanal an das On-board Computer Frontend-Modell. Dort wird es über einen gemeinsamen Speicherbereich (Shared Memory) an den GMFE-Treiber der seriellen Schnittstellenkarte RS422 übertragen und über die Schnittstelle zum On-board Computer Simulator gesendet. Das Simulator-Frontend nimmt dort angekommen das Telemetriepaket entgegen und stellt es den On-board Kontrollalgorithmen zur Verfügung. Diese wiederum versenden die PCDU Telemetrie zyklisch zum Missionskontrollsystem. Das Paket nimmt dabei wieder den Weg über die serielle Schnittstelle RS422 zum On-board Computer Frontend-Modell mit dem Ziel des UHF Senders (UTX) als Beispiel. Der UHF Sender überträgt das Paket zum TM/TC Frontend Adapter. Dort wird es abermals über einen gemeinsamen Speicherbereich an ein separates Programm übertragen und schließlich von dort per TCP/IP Verbindung über den SCOS Proxy zum Missionskontrollsystem SCOS-2000 gesendet. Der Weg eines Telekommandos vom Missionskontrollsystem zur PCDU beispielsweise würde den entsprechenden Weg nur in umgekehrter Reihenfolge nehmen.

4 Darstellung der Ergebnisse

4.1 Aufbau einer hard- und softwaregestützten Entwicklungsumgebung

Bevor Simulationen von Missionsszenarien möglich sind, muss die System-Simulationsumgebung vollständig aufgebaut sein. Diese als Vorergebnis zu verstehenden Entwicklungsschritte sind in den nachfolgenden Unterkapiteln zusammengefasst.

4.1.1 Implementation der Simulator-Infrastruktur

Unter der Simulator-Infrastruktur sind diejenigen Elemente, Komponenten und Werkzeuge zu verstehen, die allgemeiner Teil des Systemsimulators bzw. eines Teststandes sind, aber nicht speziell zur Repräsentation der Satellitenkomponenten, zum Beispiel den Komponentenmodellen des Kleinsatelliten *Flying Laptop* als Anwendungsfall, benötigt werden. Diese einzelnen Punkte sind Voraussetzung für die erst anschließend mögliche Durchführung von Systemsimulationen. Bereits in den Kapiteln 2 *(Grundlagen und Basistechnologien)* und 3 *(Durchführung und Umsetzung)* dieser Dissertation wurde auf diese Technologien und ihre Implementation im behandelten Projekt näher eingegangen. An dieser Stelle werden die einzelnen Punkte nochmals ergebnisorientiert dargestellt:

- Beschaffung, Installation und Einrichtung der Server und Computer-Infrastruktur zur Entwicklung von Satelliten-Komponentenmodellen und zum Aufbau von Simulations-Teststänten
- Einrichtung des UML-Modellierungswerkzeugs und des serverseitigen UML-Repositoriums an allen Entwicklungscomputern
- Initialisierung und Konfiguration des Concurrent Versions System (CVS) auf dem Server und Konfiguration der Entwicklungscomputer als Klienten
- Installation und Konfiguration des Datenbanksystems MySQL und Bereitstellung der zentralen Projektdatenbank mit einer Vielzahl von Datentabellen auf dem Server; Einrichtung der Entwicklungscomputer zur Anbindung an die Projektdatenbank als Klienten mit Hilfe eines Datenbank-Klientenprogramms
- Einrichtung des `MoonWiki` Informationsportals für das Institut für Raumfahrtsysteme und insbesondere die Umsetzung eines Systems zur kontrollierten Zugriffsautorisierung; Bereitstellung einer zentralen Plattform zur Datenspeicherung und zum Informationsaustausch für alle Projekte des Instituts
- Schaffung des Werkzeugs `mysql2xml` zum automatisierten Erstellen von XML-Konfigurationsdateien für den Systemsimulator basierend aus Daten der Projektdatenbank
- Beschaffung, Installation, Konfiguration und Inbetriebnahme des Missionskontrollsystems SCOS-2000 der ESA zur Verwendung als zentrales Steuerorgan für den Systemsimulator und virtuellen Satellit im MDVE Projekt am Institut für Raumfahrtsysteme

4.1.2 Erstellung von Komponentenmodellen des Echtzeit-Simulators

Die Anwendung des Systemsimulators ist im Kleinsatellitenprojekt *Flying Laptop* erfolgt. Dazu sind alle funktionalen Satellitenkomponenten (siehe Abbildung 2.3 auf Seite 8) durch ein eigenständiges Softwaremodell im Systemsimulator abgebildet worden. Jedes dieser Modelle repräsentiert das exakte Verhalten der jeweiligen Satellitenkomponente zum aktuellen Zustand und unter den aktuellen Umgebungsbedingungen. Innerhalb des Echtzeit-Simulators kommunizieren die Modelle über virtuelle Leitungen auf der Bit-Ebene exakt in der Form, wie sie auch als Hardware kommunizieren würden. Die externen Schnittstellen des Softwaremodells sind gleich denen der Satellitenhardware, damit einzelne Softwaremodelle aus der Simulation herausgelöst und durch die jeweilige Hardwarekomponente ersetzt werden können. Tabelle 4.1 fasst alle erstellten Softwaremodelle zusammen und stellt dar, in welchem realisiertem Teststand der Software-Verifikations-Einrichtung sie integriert worden sind.

Tabelle 4.1: Zusammenfassung und Integrationszeitpunkt aller erstellen Softwaremodelle

Subsystem	Akronym	Modell/ Infrastruktur Element	SVF v0.1	SVF v0.5	SVF v1.0
Simulator Infra-struktur	EnvDyn	EnvDyn Propagator	X	X	X
		Thermal Propagator/ Modell		X	X
	TMTCFE	TM/TC Frontend		X	X
ACS	RW	Reaktionsrad		X	X
	MGT	Magnetorquer		X	X
	MGM	Magnetometer		X	X
	SuS	Sonnensensorsystem		X	X
	GPS	GPS Receiver		X	X
	STR	Star Tracker		X	X
	FOG	Faseroptische Kreisel		X	X
PSS	PCDU	Energieverteilungseinheit	(X)	X	X
	SPL/M/R	Solargenerator		X	X
	BATT	Batteriesystem		X	X
C&DH	OBC	On-board Computer (Einfaches Model)	(X)	X	X
	OBC	FPGA in-the-loop			X
TT&C	S-band	S-band Transmitter/ Receiver		X	X
	UHF	UHF Transmitter/ Receiver		X	X
	VHF	VHF Transmitter/ Receiver		X	X
P/L	KA	Ka-band Transmitter/ Receiver		X	X
	KU	Ku-band Transmitter		X	X
	MICS	Optisches Kamerasystem		X	X
	TICS	Thermal-Infrarot Kamerasystem		X	X
	PAMCAM	Panorama Kamera		X	X
TCS		Temperatur Sensoren/ Heizelemente		X	X

Die korrekte Funktionalität jedes Softwaremodells ist durch eine individuelle Testanwendung mit einer entsprechenden Menge von Testfällen sichergestellt worden. Die Testfälle orientieren sich dabei an der Herstellerspezifikation und Berücksichtigen im Weiteren auch die am Institut für Raumfahrtsysteme überprüften Spezifikationen der Satellitenkomponente. Auf diese Weise wird gewährleistet, dass das Softwaremodell auf die gleiche Eingabe bzw. die gleichen Umgebungsbedingungen auch mit der gleichen Antwort bzw. dem korrekten Format reagiert, wie es die Satellitenhardware nach Herstellerspezifikation oder der projektinternen Nachweise erwarten lassen. Die Ergebnisse der Modellverifikation durch die einzelnen Testfälle sind schriftlich in einem Testdokument, der so genannten „Test Documentation“, festgehalten. Zusätzlich zum Testdokument ist eine detaillierte Beschreibung über das Konzept der Modellerstellung und die jeweilige Repräsentation der Satellitenhardware in einem weiteren Dokument, der so genannten „Model Description“, angefertigt worden. So entstehen für jedes erstellte Softwaremodell zwei Dokumente.

Ein besonderes Modell stellt die Einbindung des FPGA-Entwicklungsboards mit den darauf betriebenen On-board Kontrollalgorithmen in den Simulationskreislauf dar. Das simulatorseitig geschaffene Frontend `OBCdev_FE` bündelt alle Kommunikationssignale und sendet diese über die serielle Schnittstelle RS422 an das FPGA-Entwicklungsboard, wo sie innerhalb des Simulator-Frontends `SIM_FE` wieder zu den einzelnen Kommunikationsroutinen der On-board Kontrollalgorithmen weitergeleitet werden. Die Bündelung der Kommunikationssignale über eine einzelne Schnittstelle vereinfachte die Einbindung des On-board Kontrollalgorithmen in den Simulationskreislauf zum aktuellen Entwicklungsstand erheblich. Dadurch sind Simulationen mit den On-board Kontrollalgorithmen und geschlossenem Simulationskreislauf erst möglich geworden und der damit generierte Teststand trägt einen wesentlichen Anteil zur Entwicklung und funktionalen Verifikation der Kontrollalgorithmen bei.

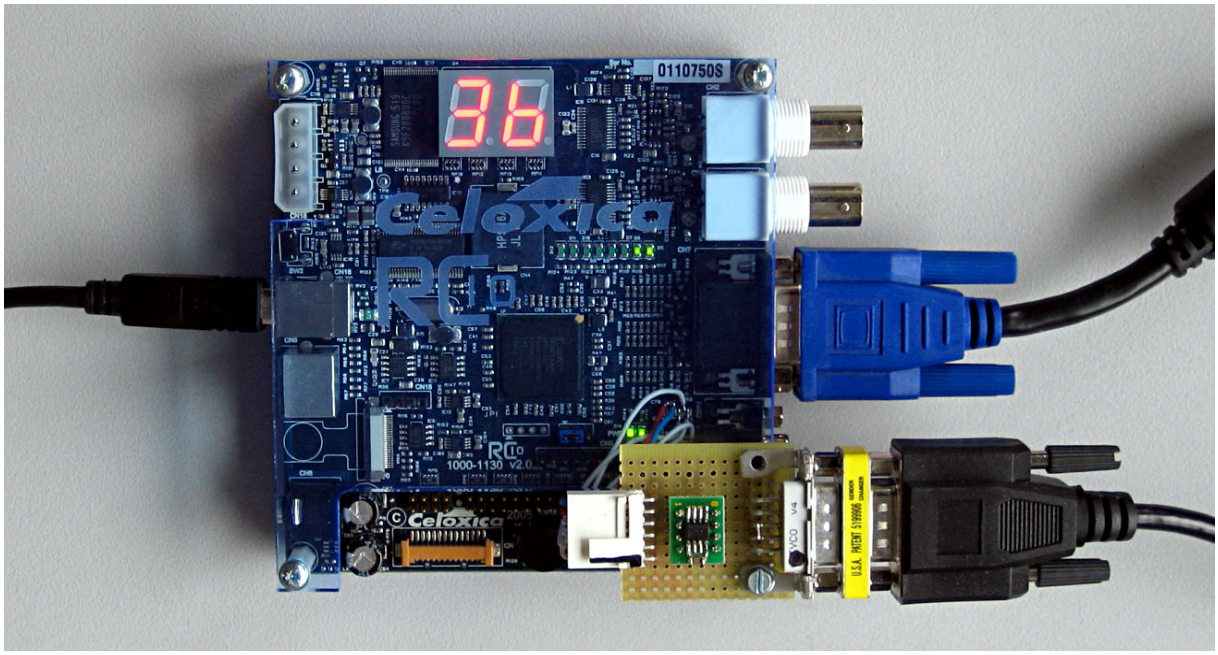


Abbildung 4.1: Das FPGA-Entwicklungsboard „RC10“ der Firma Celoxica mit modifizierter serieller Schnittstelle

Abbildung 4.1 zeigt das FPGA-Entwicklungsboard „RC10“ der Firma Celoxica mit modifizierter Schnittstelle, um die Datenübertragung über den schnelleren Standard der seriellen Schnittstelle RS422 im Dauerbetrieb bei 921600 Baud zu ermöglichen.

4.2 Simulationen von Missionsszenarien im Hinblick auf Lageregelung, Thermal- und Energiemanagement

Sämtliche in den folgenden Abschnitten erläuterten Simulationsergebnisse sind unter Echtzeit-Bedingungen und mit Einbindung der On-board Kontrollalgorithmen auf dem oben dargestellten FPGA-Entwicklungsboard entstanden. Der Entwicklungsstand der Kontrollalgorithmen ist zum Zeitpunkt der Anfertigung dieser Dissertation noch nicht sehr weit fortgeschritten und umfasst neben den Lageregelungsalgorithmen für die Betriebsmodi `SAFE` und `DETUMBLE` nur wenige weitere Funktionalitäten. Dennoch konnten bereits einfache Missionsszenarien wie der nominelle stationäre Betrieb im `SAFE` Mode, der Übergang vom `DETUMBLE` Mode in den `SAFE` Mode oder ein Separationsszenario nach dem Raketenstart und dem Aussetzen des *Flying Laptop* vom Orbittransferfahrzeug in seinen Erdorbit simuliert werden.

Nach der Vorstellung der Simulationsergebnisse in den Betriebsmodi `DETUMBLE` und `SAFE`, unter den Gesichtspunkten der Leistung der Lageregelungsalgorithmen werden komplette Missionsszenarien unter Gesichtspunkten der Energie- und damit verknüpft der Thermal-Balance vorgestellt. Die Strategie der Energie-Balance Simulationskampagne kann wie folgt beschrieben werden: In der ersten Simulationsaufgabe wird der Referenzorbit mit den ungünstigsten Bedingungen für das Energieversorgungssystem als so genannter *Worst Case* (Ungünstigster Fall) identifiziert. Alle anschließend durchgeführten Simulationen erfolgen dann im *Worst Case* Referenzorbit. Die weiteren Simulationen haben zum Ziel die Energie-Balance oder Überlebensgrenzen für die drei unterschiedlichen Solarpaneel-Konfigurationen in den Betriebsmodi `SAFE` und `IDLE` zu verifizieren. Schließlich werden Simulationen der Separation vom Orbittransferfahrzeug und ersten Erdorbits (Launch and Early Operations Phase, LEOP) durchgeführt, um verlässliche Prozeduren für das sichere Entfalten der Solarpaneele, das Aufladen des Batteriesystems und das Überführen des Satelliten in einen sicheren Betriebszustand zu identifizieren und verifizieren.

Die Interaktion von unterschiedlichen Satellitenkomponenten und ihre Abhängigkeit von aktuellen Umgebungsbedingungen, wie der Temperatur oder der relativen Lage des Satelliten zur Sonne, erfordern die Anwendung eines Systemsimulators für diese Aufgaben. Mit einfachen Berechnungen und nur auf einzelne Subsysteme bezogene Simulationen lassen sich die hier mit einem systemweiten Simulator erzielten Ergebnisse nicht vergleichen. Sie enthalten charakteristische Unterschiede, die zu benachteiligtem Design des Satellitensystems führen können.

Das Energieversorgungssystem interagiert sowohl intern mit dem Batteriesystem und dem Solargenerator als auch extern mit den restlichen Komponenten des Satelliten. Der Batterieladezustand gibt die Batteriespannung vor und damit auch das aktuelle Leistungspotential des Solargenerators bzw. der Solarzellen, die einer Spannung-Strom-Temperatur-Abhängigkeit folgen. Der maximale Batterieladestrom ist auf ein Fünftel (C/5) der Batteriekapazität begrenzt, um eine hohe Lebensdauer der Batterie zu gewährleisten. Erreicht die Batterie beim Aufladen ihre maximale Batteriespannung, wird diese im weiteren Verlauf der Batterieladung konstant gehalten. Dazu muss der Batterieladestrom durch Kurzschließen einzelner Solarzellenstrings oder Dissipation der überschüssigen Energie entsprechend reduziert werden. Die Energiedissipation findet an den Solarpaneelen statt und führt natürlich zu einer weiteren Erwärmung derselben. Weiterhin hängen die Leistung des Solargenerators und des Batterie-

systems in nicht vernachlässigbarer Weise von ihrer aktuellen Temperatur ab. Die Batterie muss aktiv innerhalb eines engen Temperaturkorridors zur Wahrung ihrer Funktionalität und Lebensdauer gehalten werden. Zur Kühlung ist die Batterie mit einem Wärmeradiator verbunden und bei Abkühlung unter 8° Celsius aktiviert sich zur Erwärmung ein Heizelement an der Batterie. Die Deaktivierung des Heizelements erfolgt beim Erreichen von 14° Celsius Batterietemperatur. Die temperaturabhängige Aktivität des Batterieheizelementes und der weiteren Heizelemente erhöht natürlich interaktiv den Energieverbrauch des Satelliten.

Die Betriebsmodi des Lageregelungssystems

Die in dieser Dissertation behandelten Betriebsmodi des Lageregelungssystems sind der `DETUMBLE` Mode, der `SAFE` Mode und der `IDLE` Mode. Abbildung 4.2 unten veranschaulicht geltende Bedingungen an den Bewegungszustand und die relative Lage zur Sonne für die Betriebsmodi `SAFE` und `IDLE` in einer graphischen Darstellung.

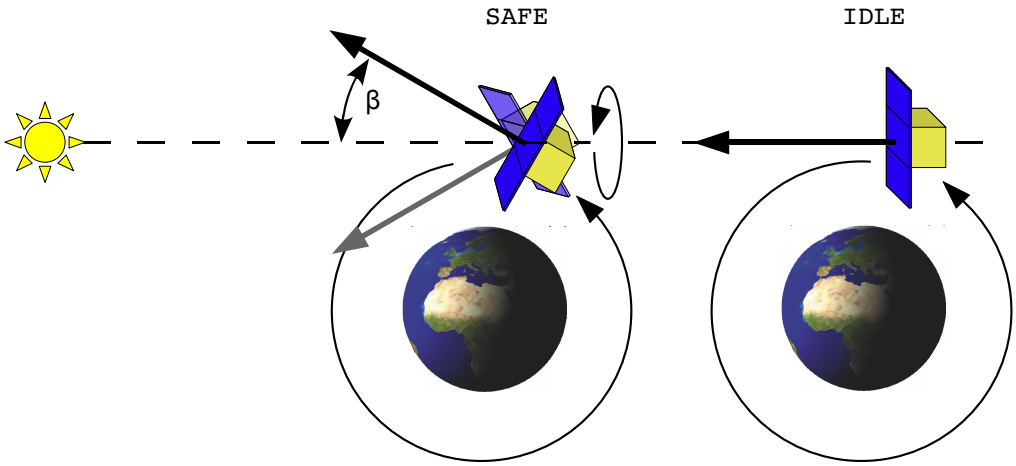


Abbildung 4.2: Bedingungen des Bewegungszustandes und der relativen Lage des Satelliten Flying Laptop zur Sonne in den Betriebsmodi `SAFE` und `IDLE`

Den Bewegungszustand und die relative Lage des Satelliten zur Sonne betreffend gelten für die angesprochenen Betriebsmodi die folgenden Bedingungen:

- Der `DETUMBLE` Mode dient der Abbremsung und Stabilisierung der Rotationsbewegungen des Satelliten in allen drei Achsen. Wenn eine definierte Drehrate in allen Achsen unterschritten wird, wechselt der Satellit automatisch vom `DETUMBLE` Mode in den `SAFE` Mode.
- Der `SAFE` Mode ist dem Namen nach ein sicherer Betriebsmodus, der unter allen Umständen ein zuverlässiges Überleben des Satelliten gewährleistet. Im `SAFE` Mode rotiert der *Flying Laptop* um seine Achse mit dem größten Trägheitsmoment und richtet diese Rotationsachse zur Sonne aus. Der Winkel β bezeichnet dann die Abweichung von der optimalen Ausrichtung der Solarpaneele zur Sonne.
- Im `IDLE` Mode befindet sich der Satellit, wenn keine Fehlfunktionen vorliegen, alle Systeme nominell arbeiten und er auf Kommandos vom Boden wartet. Der `IDLE` Mode, ist wie der Name schon andeutet, eine Wartestellung in welcher der Satellit dreiachsenstabilisiert und mit den Solarpaneelen optimal zur Sonne ausgerichtet ist. Um die Z-Achse des Satelliten ist dieser so gedreht, dass die Erde möglichst nicht das Gesichtsfeld der Star Tracker abdeckt.

Die Referenzorbits im Kleinsatellitenprojekt *Flying Laptop*

Zum derzeitigen Projektstand ist noch nicht bekannt, in welchem sonnen-synchronen Orbit der *Flying Laptop* einmal ausgesetzt wird. Daher sind zu Simulationszwecken im Projekt sieben Referenzorbits bestimmt worden, die den möglichen Bereich des späteren realen Orbits abdecken. Ein stabiles Verhalten bzw. ein sicheres Funktionieren des Satelliten muss folglich in allen Referenzorbits gegeben sein, um den zuverlässigen Betrieb des Satelliten unter den später auftretenden realen Bedingungen zu gewährleisten.

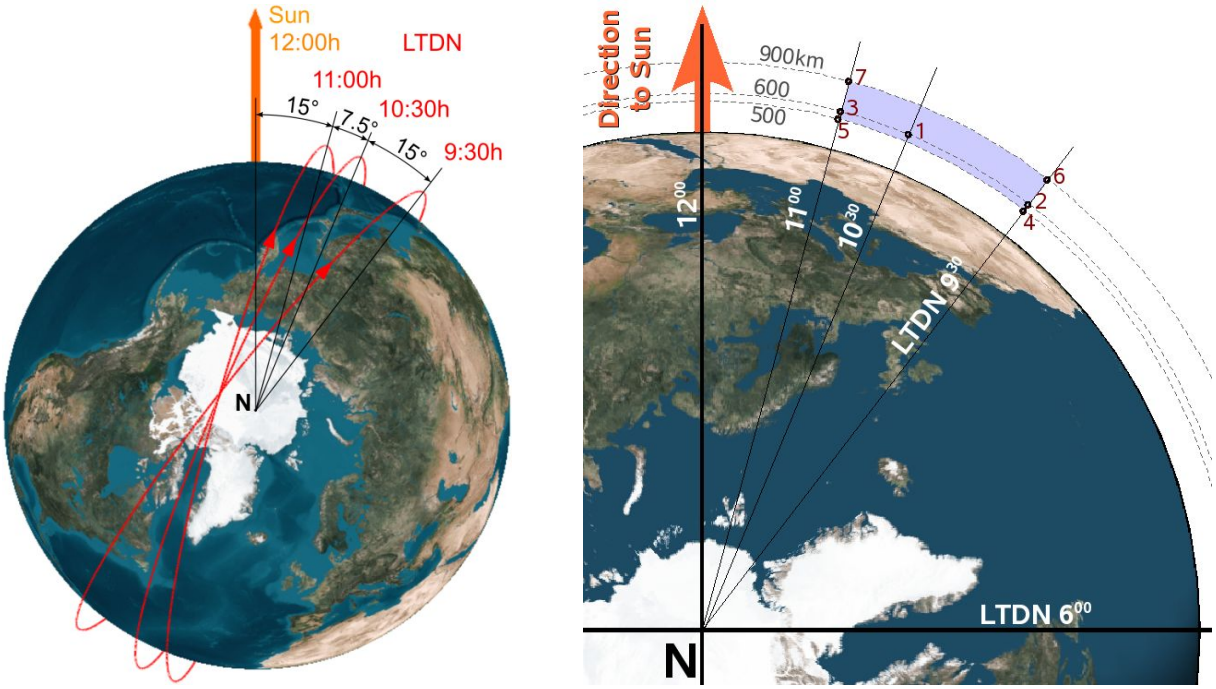


Abbildung 4.3: Maßstäbliche Illustration der geometrischen Lage der Referenzorbits

Die geometrische Lage und Orbithöhe der sieben markierten Referenzorbits ist maßstabsgetreu in Abbildung 4.3 dargestellt. Die direkten Einflussgrößen des Satellitenorbits auf das Energiemanagement sind die Orbithöhe und die Lage des absteigenden Knotens (Local Time of Descending Node, LTDN), die beide maßgeblich das Verhältnis der Zeit des Satelliten in der Sonne zu der Zeit im Erdschatten auf seinem Orbit vorgeben.

Dieses Verhältnis wiederum stellt einen charakteristischen Parameter des Referenzorbit in Bezug auf das Energiemanagement dar. Die Dauer, die der Satellit im jeweiligen Orbit im Erdschatten verbringt, repräsentiert eine absolute verbrauchte Energiemenge, die das Energieversorgungssystem bereitstellen muss. Während der sonnenbeleuchteten Phase des jeweiligen Orbits muss diese verbrauchte Energiemenge anschließend durch den Solargenerator wieder aufgeladen werden. Das Verhältnis zwischen der Zeit des Satelliten in der Sonne und der Zeit im Erdschatten repräsentiert also eine von der Lage des Satellitenorbits abhängige Wiederaufladefähigkeit der Solargeneratoren. Je größer das Verhältnis und damit die Wiederaufladefähigkeit, desto mehr Energie wird durch den Solargenerator zur Aufladung der Batterie bereitgestellt. In der nachfolgenden Tabelle 4.2 sind für die jeweiligen Referenzorbits die Verweildauer des Satelliten in der Sonne, im Erdschatten und im Halbschatten aufgeführt. Bei der Berechnung des Verhältnisses wurde die geringe Zeit im Halbschatten vernachlässigt.

Tabelle 4.2: Verweildauer des Satelliten in der Sonne, im Erdschatten und im Halbschatten in den jeweiligen Referenzorbits

Referenz-orbit	T_{Orbit}	Mittelwerte						Verhältnis Sonne / Erdschatten
		Sonne		Erdschatten		Halbschatten		
	[min]	[s]	[%]	[s]	[%]	[s]	[%]	
1	96,69	3742,4	64,43	2047,3	35,25	9,4	0,32	1,83
2	96,69	3873,3	66,68	1912,9	32,93	11,2	0,38	2,02
3	96,69	3707,5	63,83	2083,1	35,86	8,9	0,31	1,78
4	94,63	3712,7	65,31	1950,1	34,31	10,8	0,38	1,90
5	94,63	3565,5	62,72	2101,4	36,97	8,7	0,31	**1,70**
6	102,99	4340,7	70,17	1821,0	29,44	12,3	0,40	**2,38**
7	102,99	4120,1	66,6	2047,1	33,09	9,6	0,31	2,01

Die Solarpaneel-Konfigurationen des *Flying Laptop*

Während des Starts der Rakete und auch während der Separation des *Flying Laptop* vom Orbittransferfahrzeug sind die Solarpaneele an den Satellitenkörper geklappt. Ein eigens konstruierter Niederhaltemechanismus sorgt für einen sicheren Halt der Solarpaneele in dieser Stellung. Erst wenn der Entfaltungsmechanismus der Solarpaneele aktiviert wird und durch Aufheizung kleiner Heizelemente die Schmelzschnur durchgeschmolzen ist, kann die Federkraft des Entfaltungsmechanismus ihre Kraft freisetzen und die Solarpaneele nach außen drücken. Die Aktivierung des Entfaltungsmechanismus wird entweder durch ein explizites Kommando von der Missionskontrolle, autonom beim Übergang in den `SAFE` Mode oder ebenfalls autonom nach Ablauf einer Wartezeit beginnend bei der Separation initiiert.

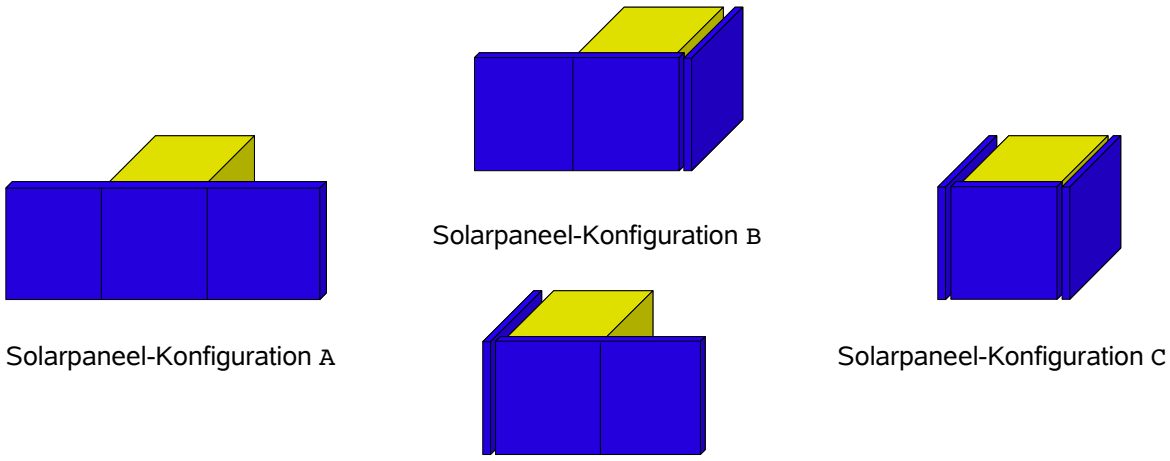


Abbildung 4.4: Die eingeführten Solarpaneel-Konfigurationen `A`, `B` und `C`

Zur Beschreibung der möglichen Stellungen der Solarpaneele sind, wie in Abbildung 4.4 illustriert, die drei Solarpaneel-Konfigurationen `A`, `B` und `C` eingeführt worden. Konfiguration `C` bezeichnet den Startzustand, wenn beide Solarpaneele noch am Satellitenkörper liegen. Konfiguration `B` identifiziert anschließend den Zustand mit einem aufgeklappten Solarpaneel und schließlich stellt Konfiguration `A` den vollständig entfalteten Zustand der Solarpaneele dar.

4.2.1 Der `DETUMBLE` Mode des Lageregelungssystems

Unter Einbindung der bereits entwickelten On-board Kontrollalgorithmen auf dem FPGA-Entwicklungsboard in die Simulationsumgebung lassen diese sich bereits auf ihre Funktionalität und Effizienz testen. So ist dem Satellit hier zu Beginn der Simulation eine Rotationsbewegung aufgeprägt worden, um die Leistung des Stabilisierungsalgorithmus im `DETUMBLE` Mode zu testen. Der anfängliche Bewegungszustand ist hier als Rotationsbewegung um 0,08 rad/sek (entspricht ~4,6°/sek) in allen drei Achsen gewählt, d. h. der Betrag der Rotationsgeschwindigkeit beträgt 0,14 rad/sek, das entspricht ungefähr 8 Grad pro Sekunde.

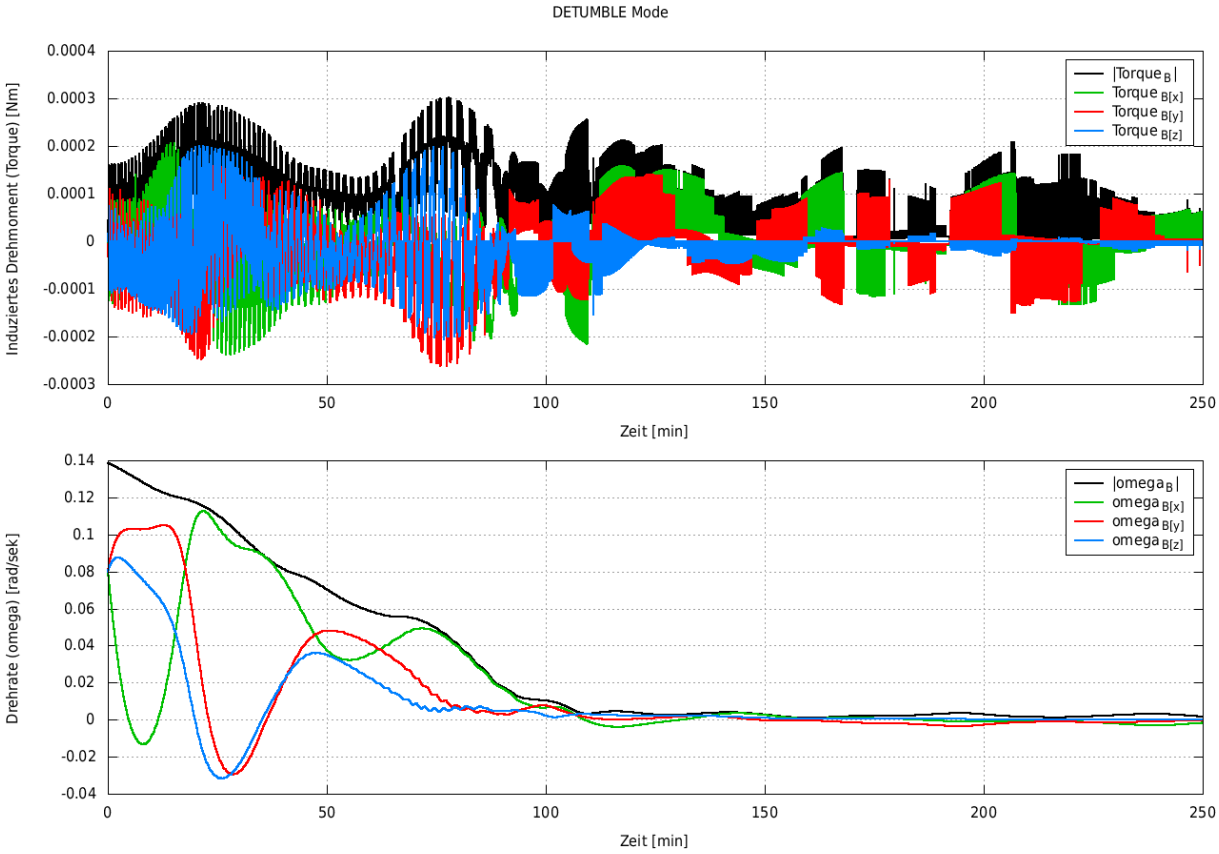


Abbildung 4.5: Induziertes Drehmoment und resultierende Drehraten im `DETUMBLE` Mode

Das Ergebnis der Simulation zeigt nun Abbildung 4.5. Dargestellt sind die durch die Magnetorquer induzierten Drehmomente zur Abbremsung der Drehbewegung des Satelliten und die daraus resultierenden Drehraten ω im körperfesten Koordinatensystem des Satelliten. Im unteren Diagramm ist unschwer zu erkennen, dass die Drehraten um die einzelnen Achsen bei der anfänglich aufgeprägten Drehrate von 0,08 rad/sek beginnen und innerhalb von ungefähr 100 Minuten nahezu vollständig abgebaut werden. Im weiteren Verlauf pendeln die Drehraten mit geringen Amplituden um die Nullpunktslage. Eine vollständige Reduktion der Drehrate bis auf den Wert Null ist aus technischen Gründen nicht möglich, da die Magnetorquer als Stellglieder keine beliebig kleinen Drehmomente erzeugen können, sondern ein feststehendes kleinstes Inkrement besitzen. Daraus resultiert die oszillierende Bewegung der Drehraten um die Nullpunktslage. Die zu jedem Zeitpunkt angreifenden Störmomente in der Weltraumumgebung (z. B.: Erdmagnetfeld, Erdgravitation und aerodynamische Momente) machen zudem ein ständig aktive Drehratenreduktion erforderlich.

Dieses einfache Simulationsergebnis zeigt durch den Verlauf des Parameters der Drehrate, dass die Lageregelungsalgorithmen des DETUMBLE Mode qualitativ richtig in den On-board Kontrollalgorithmen implementiert sind. Weiterhin ist auch die inhaltliche Kommunikation zwischen der beteiligten Satellitenhardware und den Algorithmen mit Hilfe des Systemsimulators verifiziert worden.

Im DETUMBLE Mode bestimmen die Lageregelungsalgorithmen die aktuellen Drehraten aus der Änderungsrate des Magnetfeldvektors. Da das Zeitintervall zwischen zwei Magnetfeldmessungen der Magnetometer klein ist, kann die Änderung des Erdmagnetfeldes selbst vernachlässigt werden und die Änderung zwischen den beiden Messungen ist allein auf die Drehbewegung des Satelliten zurückzuführen. Zur Reduktion dieser Drehrate wird dann ein Stellmoment in Richtung und Betrag bestimmt und ein entsprechendes Kommando an die Magnetorquer gesendet. Wenn, wie in der Simulation oben dargestellt, die Drehrate erwartungsgemäß abnimmt, ist sichergestellt, dass in den Lageregelungsalgorithmen keine Vorzeichenfehler vorhanden sind. Die Korrektheit des Simulators und der darin implementierten Komponentenmodelle ist bereits in den früher erwähnten Einzelmodelltests auch in dieser Hinsicht analytisch und numerisch verifiziert worden.

4.2.2 Der SAFE Mode des Lageregelungssystems

Der SAFE Mode des Satelliten sichert aus energetischer Perspektive ein Überleben des Satelliten unter Weltraumbedingungen. Konkret bedeutet dies, das Ausrichten der Solarpaneele in Richtung der Sonneneinstrahlung zur Erzeugung einer positiven Energiebilanz, d. h. einer betragsmäßig höheren Energieerzeugung durch die Solarzellen als vom gesamten Satelliten im gleichen Zeitraum verbraucht wird. Hier müssen auch die temperaturabhängigen Betriebszeiten von Heizelementen für das Batteriesystem berücksichtigt werden, die im SAFE Mode durch den insgesamt niedrigen Energieumsatz häufig aktiv sind.

Gleichzeitig ist die Lageregelung des Satelliten so einfach und damit so fehlerunanfällig wie möglich gestaltet. Im Falle des Kleinsatelliten *Flying Laptop* ist eine rotationsstabilisierte Lagekontrolle um die Achse mit dem größten Trägheitsmoment gewählt worden. Dabei wird die Rotationsachse so zur Sonne ausgerichtet, dass die Solarzellen auf den Solarpaneelen zur Sonne orientiert sind. Unter diesen Bedingungen beträgt der Winkel β_{Sun} zwischen der Sonneneinstrahlrichtung und dem Normalenvektor der Solarpaneele im Mittel ca. 38°. Für die Realisierung dieser Lagekontrolle werden nur die drei am wenigsten komplexen Komponenten des Lageregelungssystems verwendet: Die Magnetometer, die Magnetorquer und das Sonnensensorsystem. Die aktive Lagekontrolle, d. h. das Aufprägen von Stellmomenten erfolgt nur im sonnenbestrahlten Teil des Erdorbits außerhalb des Erdschattens, wenn das Sonnensensorsystem einen Sonnenvektor liefert. Die Rotation um die Achse mit dem größten Trägheitsmoment sorgt in der kürzeren Zeitspanne im Erdschatten für die Einhaltung der aktuellen Satellitenlage.

Der Anfangszustand der im Folgenden vorgestellten Simulationsergebnisse im SAFE Mode des Satelliten ist immer identisch. Der Satellit befindet sich auf einem der definierten Referenzorbits in einer beliebigen relativen Lage zur Sonne und besitzt keine anfängliche Drehrate, d. h. er befindet sich im körperfesten Koordinatensystem des Satelliten zum Zeitpunkt Null der Simulation in Ruhe. Sobald die Lageregelungsalgorithmen des SAFE Mode aktiv werden, beginnen sie damit, den Satelliten mit der Rotationsachse des größten Trägheitsmoments zur Sonne auszurichten und die Rotationsbewegung aufzubauen. Letzterer Prozess, der Aufbau der Rotationsbewegung, dauert mehrere Erdorbits.

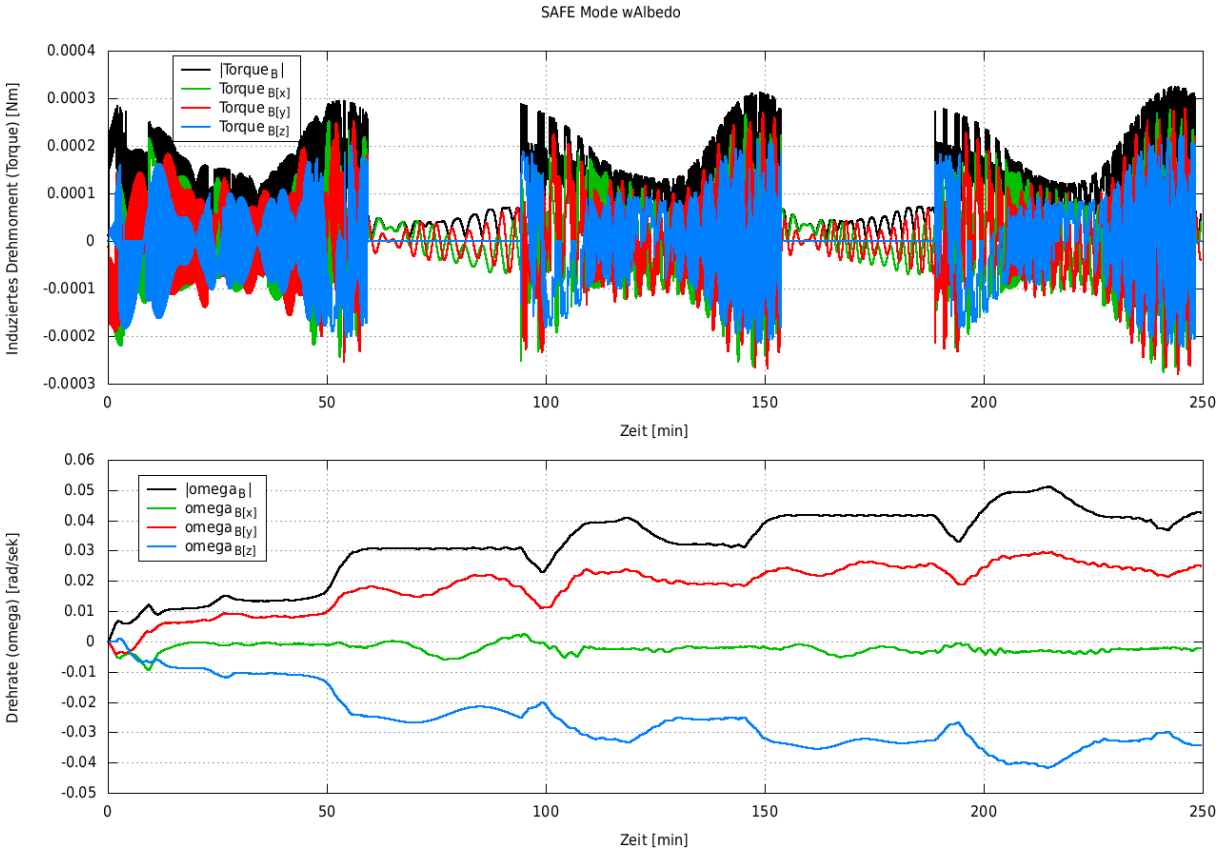


Abbildung 4.6: Induziertes Drehmoment und resultierende Drehraten im `SAFE` Mode

Die geschilderten Prinzipien des Lageregelungsalgorithmus im `SAFE` Mode lassen sich aus den Simulationsergebnissen in Abbildung 4.6 und Abbildung 4.7 einfach herauslesen. Während im sonnenbestrahlten Teil des Erdorbits aktiv Stellmomente durch die Magnetorquer aufgeprägt werden, fehlen diese Stellmomente im Erdschatten. Im oberen Diagramm der induzierten Drehmomente ist dies deutlich zu erkennen. Befindet sich der Satellit innerhalb des Erdschattens sind betragsmäßig kleine Momente erkennbar, die auf das Eigendipolmoment des Satelliten bezogen auf das Erdmagnetfeld zurückzuführen sind.

Die Grafik für den Verlauf der Drehraten zeigt diesen über der Zeit auf. Man erkennt den sukzessiven Aufbau der Rotationsbewegung um die Achse mit dem größten Trägheitsmoment in der Sonnenphase und das Halten der Drehrate in der Schattenphase. Zum Ende der dargestellten Simulationszeit, nach fast drei Erdorbits, ist die maximal angestrebte Drehrate erreicht.

Ziel der Lagekontrolle im `SAFE` Mode ist, wie bereits erwähnt, die Orientierung der Solarpaneele zur Sonne. Dabei spielt die Lage der Achse mit dem größten Trägheitsmoment eine dominante Rolle, beeinflusst ihre Lage relativ zur Z-Achse des Satelliten doch entscheidend die für die Energiegeneration optimale Sonnenausrichtung. Je eher die Achse mit dem größten Trägheitsmoment der Z-Achse des Satelliten entspricht, desto besser ist die Ausrichtung der Solarpaneele zur Sonne.

Der Orientierungswinkel β_{Sun} der Solarpaneele zur Sonne resultiert aus der aktiven Lagekontrolle des Satelliten und ist analog zu den anderen Diagrammen über der Simulationszeit in Abbildung 4.7 auf der nächsten Seite dargestellt. Im Diagramm ist weiterhin das Intervall der Sonnenbestrahlung auf dem Erdorbit durch die Strahlungsintensität in gelb markiert.

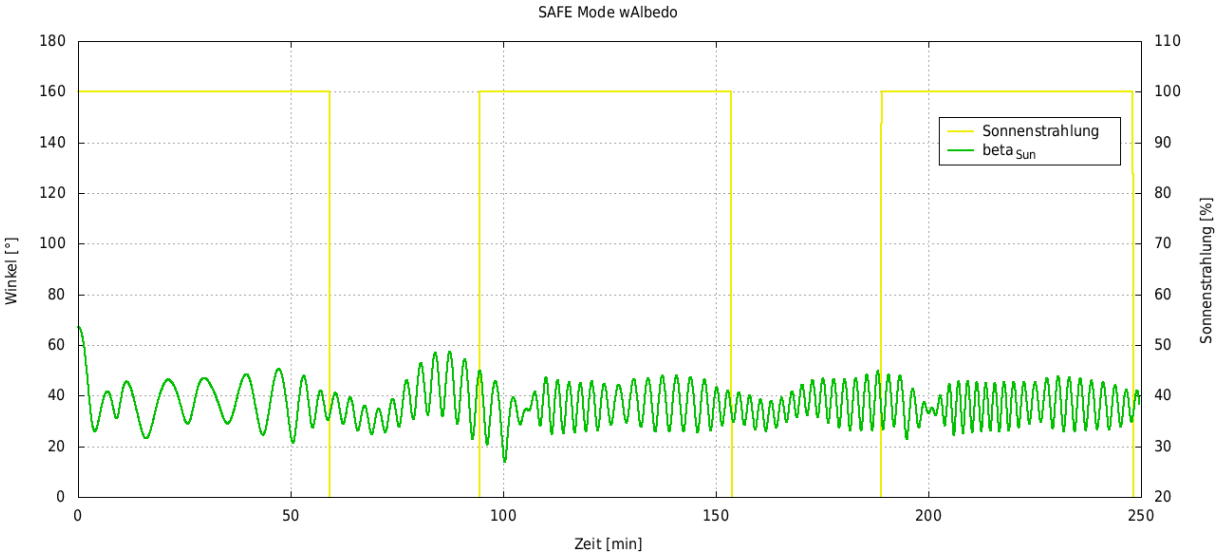


Abbildung 4.7: Orientierungswinkel β_{Sun} der Solarpaneele zur Sonne im `SAFE` *Mode*

Der noch unregelmäßige Verlauf in der Sonnenorientierung β_{Sun} während der ersten sonnenbestrahlten Phase ist durch die gleichzeitige Ausrichtung zur Sonne und Aufbringung der Rotationsbewegung bedingt. Bereits ab der zweiten sonnenbestrahlten Phase ist die oszillierende Bewegung um die Achse der optimalen Sonnenorientierung gleichmäßiger. Die noch verhältnismäßig hohe Ungenauigkeit bei der Sonnenorientierung, d. h. die große Amplitude der oszillierenden Bewegung resultiert aus der Ungenauigkeit des Sonnensensorsystems, welches bei der Bestimmung des Sonnenvektors durch die Albedostrahlung der Erde gestört wird. Die Algorithmen, die in den On-board Kontrollalgorithmen aus den Strömen des Sonnensensorsystems den Sonnenvektor berechnen, können den Effekt der Albedostrahlung nicht direkt isolieren und dadurch keinen genauen Sonnenvektor bestimmen. Demzufolge muss die Lageregelung des Satelliten mit dieser Ungenauigkeit zurecht kommen.

Erste Optimierungen des Lageregelungsalgorithmus im `SAFE` Mode

Abbildung 4.8: Vorläufige Optimierungen des `SAFE` *Mode Lageregelungsalgorithmus*

Im Folgenden wird ein erster Schritt zur Optimierung der Lageregelungsalgorithmen im `SAFE` Mode vorgestellt, der Ansätze zur Berücksichtigung des Effektes der Albedostrahlung enthält. Zum Zeitpunkt der Anfertigung dieser Dissertation waren keine weiteren Verbesserungen der Lageregelungsalgorithmen zum Test durch den Systemsimulator verfügbar.

Abbildung 4.8 zeigt den optimierten Lageregelungsalgorithmus (blau) in Relation zum ursprünglichen Algorithmus (grün) und zu einer Simulation ohne Berücksichtigung der Albedostrahlung im Sonnensensorsystem (rot). Vernachlässigt man den Effekt der Albedostrahlung im Sonnensensorsystem, so sorgen die Lageregelungsalgorithmen für eine nahezu perfekte Ausrichtung der Achse des größten Trägheitsmoments zur Sonne. Dagegen zeigen die optimierten Algorithmen zwar charakteristische Unterschiede gegenüber den ursprünglichen Algorithmen, aber keine deutlichen Verbesserungen. Zum Ende der sonnenbestrahlten Phase wird die Schwingungsamplitude sogar größer.

Die Durchführung dieser vergleichenden Simulationen zeigt deutlich den dringenden Bedarf an solchen Tests zur Optimierung bzw. Verifizierung der entwickelten Algorithmen. Aufgrund der geringen Komplexität der in der Entwicklung der Lageregelungsalgorithmen verwendeten Testumgebung sind diese Unzulänglichkeiten bisher nicht aufgefallen. Der Systemsimulator MDVE jedoch ist so konfiguriert, dass er eine Vielzahl von unterschiedlichen Tests ohne aufwendige Änderungen jederzeit in voller Komplexität durchführen kann. Somit steht einer Entwicklung und Optimierung der Lageregelungsalgorithmen und sukzessiven Tests zur Verifikation und analytischen Dokumentation nichts im Wege.

4.2.3 Identifikation des Referenzorbits mit den ungünstigsten Bedingungen für das Energieversorgungssystem

Das Ziel der hier vorgestellten Simulationen ist die Identifikation desjenigen Referenzorbits mit den ungünstigsten Bedingungen für das Energieversorgungssystem. Abbildung 4.3 auf Seite 74 illustriert den durch die Referenzorbits aufgespannten Bereich, für den das Energieversorgungssystem ausgelegt werden muss. Die Einflüsse der variierenden Orbitparameter auf das Verhalten des Energieversorgungssystems sind höhere (Orbit mit niedriger Orbithöhe) oder niedrigere (Orbit mit großer Orbithöhe) thermische Lasten und die Dauer der Phasen im Erdschatten bzw. im sonnenbestrahlten Bereich des Orbits.

Anfangsbedingungen der Simulationen

Die Solarpaneel-Konfiguration `C` ist für diese Simulationen gewählt worden, da der Einfluss der Orbitparameter auf das Verhalten des Energieversorgungssystems in dieser Konfiguration, wenn beide Solarpaneele nicht entfaltet sind, am Deutlichsten zu erkennen ist. Die Simulationen werden im `SAFE` Mode des Satelliten durchgeführt, wodurch der Energiebedarf aller aktiven Satellitenkomponenten im nominellen Betrieb festgelegt ist. Eine Darstellung der relativen Ausrichtung des Satelliten zur Sonne im `SAFE` Mode gibt Abbildung 4.2 auf Seite 73 wieder.

Das Verhalten der Hardware des Energieversorgungssystems wird unter den Bedingungen berücksichtigt, wie sie zum Ende der angestrebten Lebenszeit des Satelliten von zwei Jahren auftreten werden (End-of-Life, EOL). Dieses Kriterium definiert die Annahme einer Degradation der Solarzellen um 10% und eine um 11% [GS & JSB '03] reduzierte maximale Batteriekapazität des Batteriesystems. Zu Beginn der Simulation ist das Batteriesystem nahezu vollständig aufgeladen.

Vergleichskriterium der Simulationen in den Referenzorbits

Um das Verhalten und die Leistungsfähigkeit des Energieversorgungssystems im Gesamten zu bewerten, muss der aktuelle Ladezustand der Batterie (State-of-Charge, SOC) verwendet werden. Er gibt Aufschluss über die Energiebilanz eines oder mehrerer simulierter Erdorbits unter variierenden Bedingungen. Seine Charakteristik enthält alle definierenden Parameter des Energieversorgungssystems wie die verbrauchte Energie, die durch die Solarzellen neu generierte Energie, die Dauer der Phase im Erdschatten, das Verhältnis zwischen der Zeit in der Sonne und im Erdschatten, die charakteristische Temperaturen und die Aktivität von Heizelementen. Der Batterieladezustand ist definiert durch die aktuelle Batteriekapazität dividiert durch die nominelle Batteriekapazität. Im Kontext der hier durchgeführten Simulationen wird er in einer dynamischen Art und Weise verwendet, d. h. die nominelle Batteriekapazität wird in Abhängigkeit der aktuellen Batterietemperatur, ihrer bisherigen Lebenszeit und kapazitätsreduzierender Effekte [GS & JSB '03] der insgesamt bereits durchlaufenen Lade- bzw. Entladezyklen bestimmt.

Simulationsergebnisse

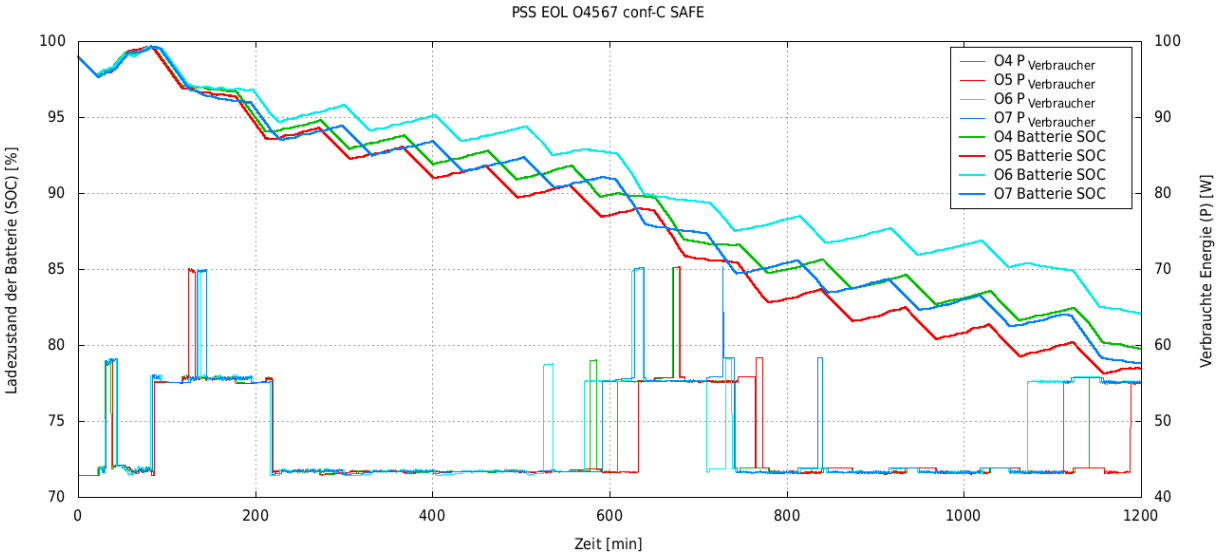


Abbildung 4.9: Einflüsse der Orbitparameter auf die Energiebilanz

Die direkten Einflüsse der Orbitparameter auf die Energiebilanz sind anhand der Verläufe für die verbrauchte Energie und den resultierenden Batterieladezustand in Abbildung 4.9 für alle vier Referenzorbits über die Zeitdauer von 1200 Minuten (entspricht ca. 12 Erdorbits) illustriert. Mit diesem Ergebnis kann der Referenzorbit mit den ungünstigsten Bedingungen für das Energieversorgungssystem bestimmt werden.

Die insgesamt verbrauchte Energie aller vier Referenzorbits ist im unteren Bereich des Diagramms dargestellt. Der Betrag variiert durch kurzzeitige Spitzenlasten als Ergebnis wiederkehrender Kontaktzeiten mit der Bodenstation. Zu diesen Zeiten wird mit höherer Datenrate kommuniziert, was einen höheren Leistungsbedarf erfordert. Die länger dauernden Spitzenlasten werden durch die Aktivität des Heizelementes des Batteriesystems verursacht.

Der Verlauf des Batterieladezustands ergibt sich anschließend aus der von den aktiven Satellitenkomponenten verbrauchten Energie und der durch die Solarzellen erzeugten elektrischen Leistung. Die gesamte, während eines Ladezyklus von den Solarzellen zur Verfügung

gestellten Leistung, ist stark von den Orbitparametern abhängig. Je größer das Verhältnis zwischen der Zeit im sonnenbestrahlten Abschnitt des Orbits und der Zeit im Erdschatten ist, desto mehr Energie können die Solarzellen liefern. Dieses Verhältnis ist für alle Referenzorbits bereits in Tabelle 4.2 auf Seite 75 angegeben und stellt die maßgebliche Größe für die Identifikation der ungünstigsten Orbitparameter dar. Dies konnte durch die hier durchgeführten Simulationen bestätigt werden. Referenzorbit Nummer `6` erweist sich als derjenige Orbit mit den besten Bedingungen für das Energieversorgungssystem während Referenzorbit Nummer `5` die schlechtesten Bedingungen aufweist. In keinem der Orbits kann die im Erdschatten verbrauchte Energie während des sonnenbestrahlten Teils des Orbits in der Solarpaneel-Konfiguration `C` aufgrund der ungenauen Ausrichtung der Solarpaneele zur Sonne wieder aufgeladen werden. Somit fällt der Batterieladezustand sukzessive ab.

Hinzuweisen ist hier auch auf die Aktivitätsperioden der Heizelemente des Batteriesystems. Dieses hat im Vergleich der Referenzorbits zwar keinen bemerkenswerten Einfluss auf die Energiebilanz, zeigt aber deutlich eine Interaktion der thermalen Zustände des Satelliten mit dem Energieversorgungssystem. Fällt die Batterietemperatur unter einen minimal erlaubten Wert, aktiviert sich das Heizelement und erwärmt das Batteriesystem bis zum Erreichen eines oberen Schwellwertes. Gerade bei der Betrachtung des `SAFE` Mode mit prinzipiell äußerst niedrigen Energieanforderungen ist die dabei verbrauchte elektrische Heizleistung nicht unerheblich und stellt eine signifikante Dimensionierungsgröße für das Energieversorgungssystem dar.

Zusammenfassung

Referenzorbit Nummer `5` ist als das ungünstigste Szenario für das Energieversorgungssystem identifiziert worden. Das Potential der Wiederaufladung der Batterie ist in Referenzorbit Nummer `5` am Geringsten, da das Verhältnis zwischen der Zeit im sonnenbestrahlten Abschnitt des Orbits und der Zeit im Erdschatten den geringsten Wert aufweist. In allen folgenden auf die Energiebilanz bezogenen Simulationen wird jetzt dieser Orbit verwendet.

4.2.4 Vergleich der Solarpaneel-Konfigurationen im `SAFE` Mode des Satelliten

Die hier vorgestellten Simulationen untersuchen, ob der `SAFE` Mode in seiner aktuellen Definition in allen Solarpaneel-Konfigurationen als stabiler Zustand aus energetischer Sicht betrachtet werden kann. Die Simulationen werden unter den ungünstigsten Bedingungen für das Energieversorgungssystem durchgeführt.

Anfangsbedingungen der Simulationen

Allen hier miteinander verglichenen Simulationen liegen die Orbitparameter des Referenzorbit Nummer `5` zugrunde und sie werden im `SAFE` Mode durchgeführt. Das Verhalten der Hardware des Energieversorgungssystems wird unter den Bedingungen zum Ende der angestrebten Lebenszeit des Satelliten von zwei Jahren berücksichtigt.

Vergleichskriterium der Simulationen verschiedener Solarpaneel-Konfigurationen

Als Vergleichskriterium der Simulationen mit den unterschiedlichen Solarpaneel-Konfigurationen `A`, `B` und `C` wird auch hier der Batterieladezustand herangezogen. Er gibt wieder Aufschluss über die Energiebilanz der simulierten Erdorbits unter variierenden Bedingungen.

Simulationsergebnisse und Zusammenfassung

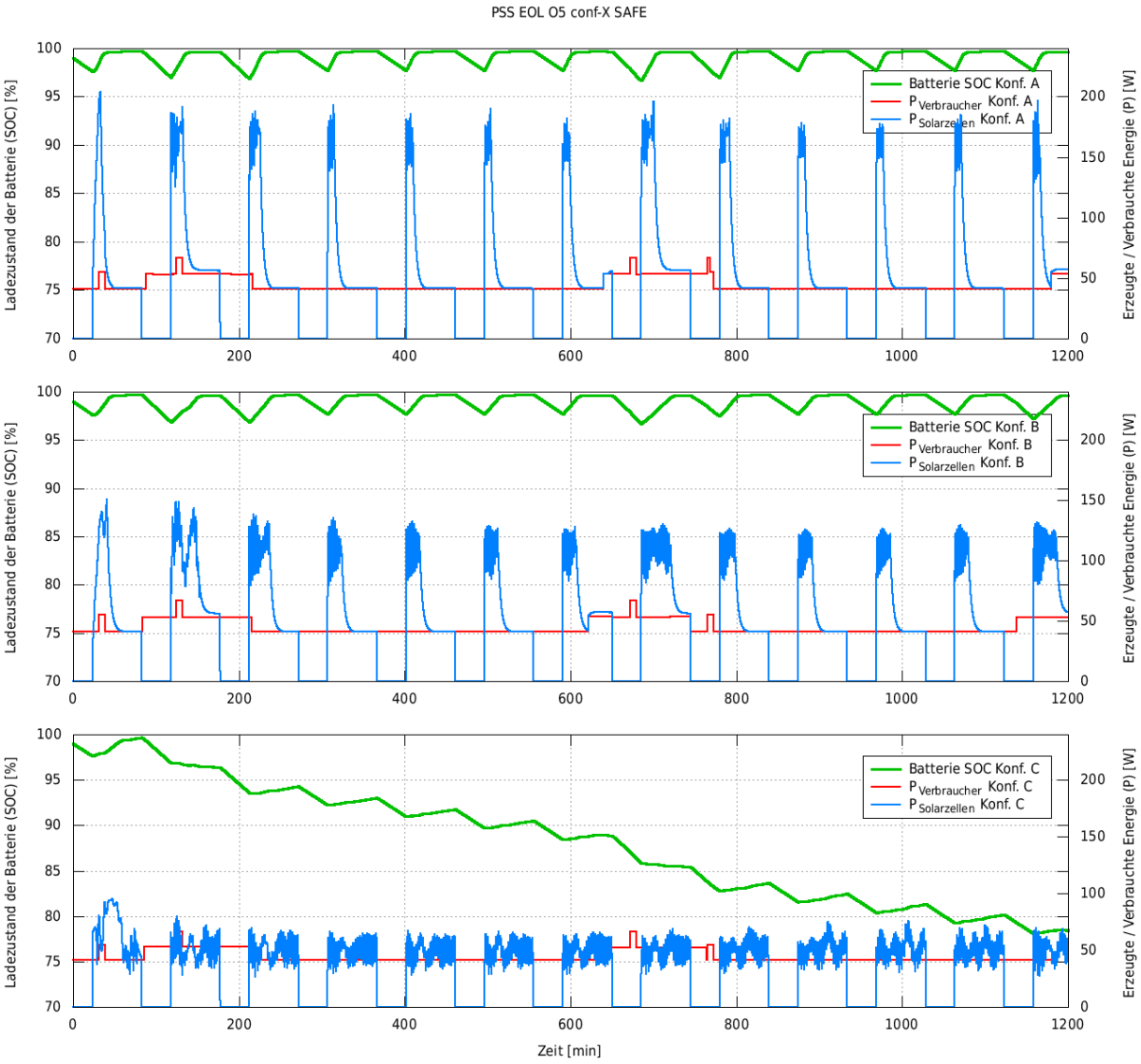


Abbildung 4.10: Energiebilanz im `SAFE` Mode in den Solarpaneel-Konfigurationen `A`, `B` & `C` unter ungünstigsten Simulationsbedingungen für das Energieversorgungssystem

Die in Abbildung 4.10 dargestellten Simulationsergebnisse bestätigen aus energetischer Sicht die Stabilität der Energiebilanz in den Solarpaneel-Konfigurationen `A` und `B` im `SAFE` Mode. In beiden Konfigurationen kann die Batterie während der Ladephase im sonnenbestrahlten Teil des Erdorbits nahezu vollständig aufgeladen werden. Der Batterieladezustand wird den Zustand 100% aufgrund von Beschränkungen der Ladeschlussspannung (End-of-Charge-Voltage, EOCV) und damit einer Begrenzung des Batterieladestromes nie erreichen. Die Ladestrombegrenzung ist in der zum Ende der Sonnenphase hin abfallenden Kurve der von den Solarzellen erzeugten Leistung $P_{Solarzellen}$ sichtbar.

In Solarpaneel-Konfiguration `C` kommt es aufgrund der ungenauen bzw. oszillierenden Ausrichtung des mittleren Solarpaneels zur Sonne nicht zu einer stabilen und damit zuverlässigen Energiebilanz im `SAFE` Mode. Hier liegt noch sichtbares Optimierungspotential der Lageregelungsalgorithmen für den `SAFE` Mode vor, um auch in Solarpaneel-Konfiguration `C` einen

energetisch stabilen und zuverlässigen Zustand zu gewährleisten. Ein längerfristiger Verbleib des Satelliten in der Solarpaneel-Konfiguration C ist zum momentanen Entwicklungszustand äußerst kritisch zu betrachten und nicht empfehlenswert.

4.2.5 Vergleich der Solarpaneel Konfigurationen im IDLE Mode des Satelliten

Die hier vorgestellten Simulationen untersuchen analog zu der vorangegangenen Simulation, ob der IDLE Mode in seiner aktuellen Definition in allen Solarpaneel-Konfigurationen als stabiler Zustand aus energetischer Sicht betrachtet werden kann. Die Simulationen werden unter den ungünstigsten Bedingungen für das Energieversorgungssystem durchgeführt.

Anfangsbedingungen der Simulationen und Vergleichskriterium

Allen Vergleichssimulationen liegen die Orbitparameter des Referenzorbit Nummer 5 zugrunde und sie werden im IDLE Mode durchgeführt. Eine Darstellung der relativen Ausrichtung des Satelliten zur Sonne im IDLE Mode gibt Abbildung 4.2 auf Seite 73 wieder. Das Verhalten der Hardware des Energieversorgungssystems wird unter den Bedingungen zum Ende der angestrebten Lebenszeit des Satelliten von zwei Jahren berücksichtigt.

Vergleichskriterium der Simulationen verschiedener Solarpaneel-Konfigurationen

Als Vergleichskriterium der Simulationen mit den unterschiedlichen Solarpaneel-Konfigurationen A, B und C wird auch hier der Batterieladezustand herangezogen. Er gibt wieder Aufschluss über die Energiebilanz der simulierten Erdorbits unter variierenden Bedingungen.

Simulationsergebnisse und Zusammenfassung

Die in Abbildung 4.11 auf der folgenden Seite dargestellten Simulationsergebnisse bestätigen aus energetischer Sicht die Stabilität der Energiebilanz in den Solarpaneel-Konfigurationen A und B im IDLE Mode. In beiden Konfigurationen kann die Batterie während der Ladephase im sonnenbestrahlten Teil des Erdorbits nahezu vollständig aufgeladen werden.

In Solarpaneel-Konfiguration C kommt es aufgrund des deutlich höheren Energieverbrauchs im Vergleich zur erzeugten Leistung der Solarzellen nicht zu einer stabilen und damit zuverlässigen Energiebilanz im IDLE Mode. Der IDLE Mode ist somit in der Solarpaneel-Konfiguration C nicht längerfristig tragbar. Sollten nach dem Start des Satelliten beide Solarpaneele nicht entfaltet werden können, muss nach einer energetisch stabilen Alternative für den IDLE Mode gesucht werden. Möglicherweise kann der Satellit dann mit einem der beiden seitlichen Solarpaneele zur Sonne ausgerichtet werden, da diese durch eine höhere Anzahl von Solarzellen im Vergleich zum mittleren Solarpaneel mehr elektrische Leistung erzeugen können. Es bleibt jedoch zu untersuchen, ob diese variierte Lagekontrolle zu einer stabilen Energiebilanz führt.

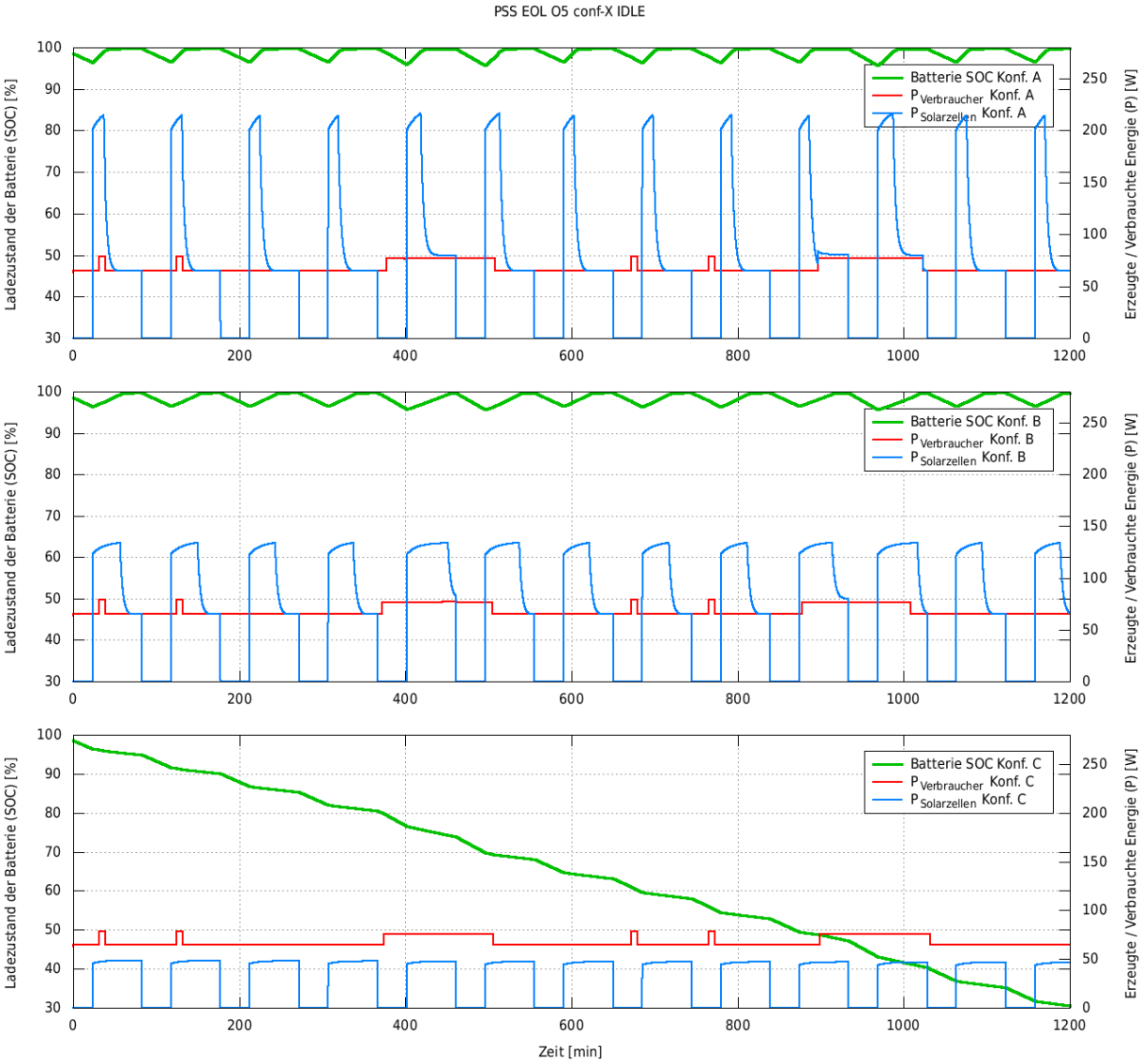


Abbildung 4.11: Energiebilanz im `IDLE` Mode in den Solarpaneel-Konfigurationen `A`, `B` & `C` unter ungünstigsten Simulationsbedingungen für das Energieversorgungssystem

4.2.6 Simulation von Separationsszenarien des Satelliten vom Orbittransferfahrzeug und seiner ersten Erdumkreisungen

Diese Simulationen dienen dazu, ein stabiles und sicheres Szenario für die Separation des Satelliten vom Orbittransferfahrzeug und seiner ersten Erdumkreisungen zu identifizieren. Als wichtigste Fragestellung wird hier untersucht, zu welchem Zeitpunkt und damit durch welches Prinzip die beim Raketenstart an den Satellitenkörper angelegten Solarpaneele nach der Separation entfaltet werden sollen bzw. müssen. Zur Diskussion steht einerseits ein vom Satelliten autonom initiiertes Entfalten der Solarpaneel beim Übergang des Satelliten vom `DETUMBLE` Mode in den `SAFE` Mode und andererseits ein entsprechendes Kommando von der Missionskontrolle. Das Kommando kann allerdings erst bei Funkkontakt des *Flying Laptop* mit der Bodenstation erfolgen und dies ist zeitlich erst später der Fall. Unter ungünstigen Umständen kann dies sogar durch ungenaue Informationen der Bahnparameter des gerade erst

gestarteten Satelliten noch erheblich verzögert werden. Die durch die Missionskontrolle initiierte Entfaltung der Solarpaneele birgt also ein zeitliches Risiko, welches der Satellit nur durch einen äußerst stabilen energetischen Zustand bis zur Entfaltung minimieren kann. Dieser energetisch stabile Zustand von der Separation bis zur Solarpaneel-Entfaltung ist nach derzeitigen Untersuchungen jedoch nicht der Fall.

Entsprechend der Untersuchungsziele sind folglich zwei charakteristisch unterschiedliche Simulationsszenarien definiert worden.

- **Szenario 1**: Nach der Separation des Satelliten vom Orbittransferfahrzeug stabilisiert der Satellit seine Lage im `DETUMBLE` Mode und geht anschließend automatisch in den `SAFE` Mode über. Bei diesem Übergang erfolgt die Entfaltung der Solarpaneele autonom vom Satelliten gesteuert.
- **Szenario 2**: Gegenüber dem ersten Szenario erfolgt das Entfalten der Solarpaneele nicht autonom vom Satelliten gesteuert beim Wechsel des Betriebsmodus in den `SAFE` Mode. Das Entfalten der Solarpaneele wird von der Missionskontrolle bei Funkkontakt im zweiten Erdorbit initiiert. Zu diesem Zeitpunkt befindet sich der Satellit bereits im `SAFE` Mode. Die Annahme eines Funkkontakts bereits im zweiten Erdorbit des Satelliten ist als optimistisch zu bewerten und bezieht mehrere auf der Erdoberfläche verteilte und zur Verfügung stehenden Bodenstationen mit ein. Ist dies nicht der Fall, kann das Entfaltungskommando erst zu einem späteren Zeitpunkt erfolgen und der Satellit verbleibt länger in der aus energetischer Perspektive kritischen Solarpaneel-Konfiguration `C`. Dies führt dann zu einer tieferen Entladung des Batteriesystems.

Anfangsbedingungen der Simulationen

Die Simulation beginnt exakt zu dem Zeitpunkt, wenn sich der Satellit vom Orbittransferfahrzeug löst. Die dann geltenden Umgebungs- und Zustandsbedingungen unterliegen äußeren Einflüssen, die durch den Aufenthalt des Satelliten unter der Nutzlastverkleidung des Orbittransferfahrzeugs und durch die Separation selbst definiert sind.

Die Temperatur des gesamten Satelliten wird unter diesen Bedingungen üblicherweise isotherm zu 20°C angenommen. Erst nach der Separation, wenn der Satelliten nicht mehr unter der Nutzlastverkleidung verborgen ist, sondern den Weltraumbedingungen ausgesetzt ist, ändert sich der thermische Zustand des Satelliten. Für das Energieversorgungssystem gelten Bedingungen wie zu Beginn der Lebenszeit der einzelnen Komponenten, d. h. sowohl das Batteriesystem als auch die Solarzellen weisen ihren nominellen und optimalen Wirkungsgrad auf. Allerdings ist das Batteriesystem nur zu ungefähr 50% geladen, da dies eine Sicherheitsauflage des Startanbieters für den Raketenstart ist.

Die Separation des Satelliten vom Orbittransferfahrzeug versetzt ihn in eine unbekannte Lageausrichtung und eine unbekannte Rotationsbewegung. Zu Simulationsbeginn wird somit eine beliebige Satellitenlage angenommen und eine angenommene Rotationsbewegung von 0,08 rad/sek (entspricht ~4,6°/sek) in jede der drei Achsen aufgeprägt, d. h. der Betrag der Rotationsgeschwindigkeit beträgt knapp 0,14 rad/sek, das entspricht ungefähr 8 Grad pro Sekunde.

Die Simulationen werden in dem für das Energieversorgungssystem ungünstigsten Referenzorbit Nummer `5` durchgeführt.

Vergleichskriterium der Simulationen von Separationsszenarien

Als Vergleichskriterium der Simulationen unterschiedlicher Separationsszenarien wird auch hier der Batterieladezustand herangezogen. Er gibt wieder Aufschluss über den Verlauf der Energiebilanz zu Beginn des Satellitendaseins im Erdorbit. Insbesondere der Zeitpunkt des Wechsels von negativer zu positiver Energiebilanz ist ein Unterscheidungsmerkmal.

Simulationsergebnisse

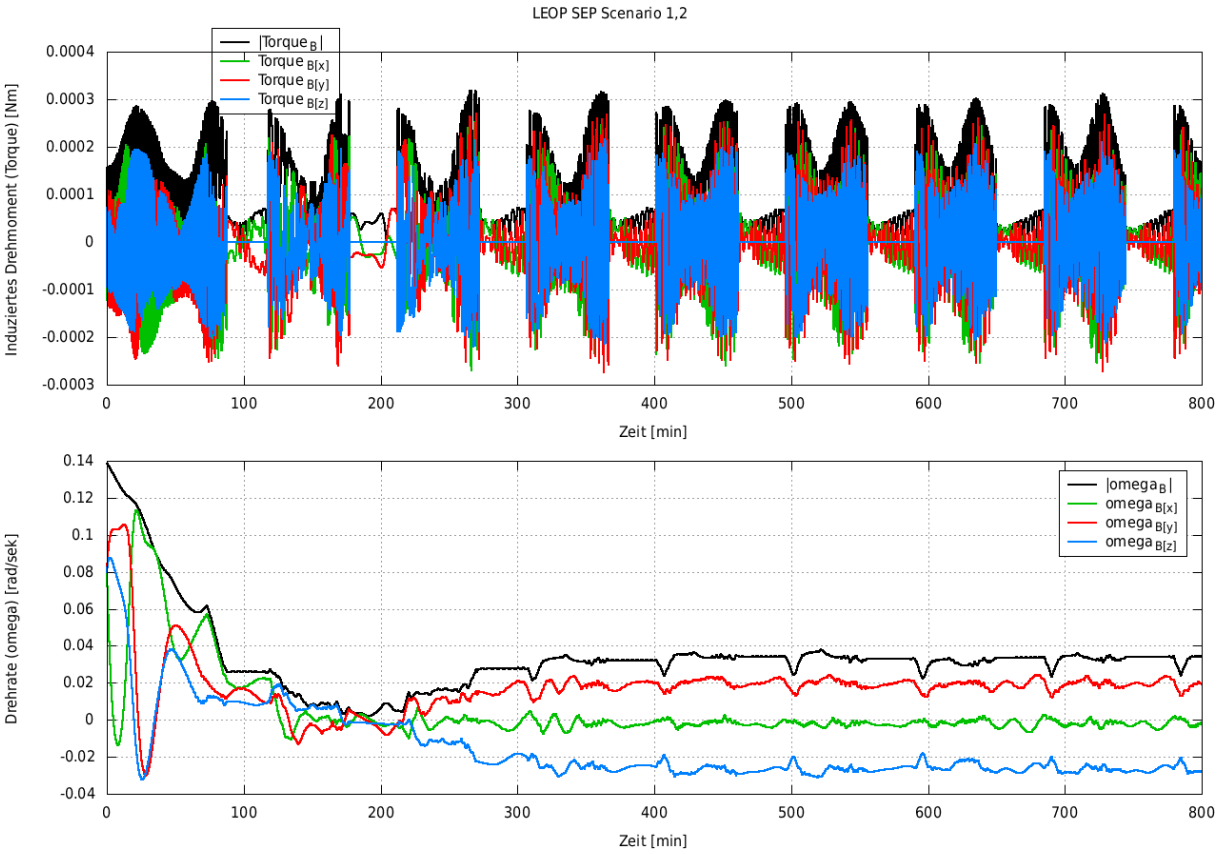


Abbildung 4.12: Induziertes Drehmoment und resultierende Drehraten für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug

Der Verlauf für das induzierte Drehmoment durch die Magnetorquer bzw. durch das Eigendipolmomentes des Satelliten und die daraus resultierenden Drehraten sind für beide Szenarien qualitativ und quantitativ identisch und in Abbildung 4.12 für die ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug dargestellt. Man erkennt deutlich die initiale Drehrate aller drei Achsen von 0,08 rad/sek am linken Rand des Diagramms. Die Drehrate wird im `DETUMBLE` Mode sichtbar sehr zügig reduziert, woraufhin beim Erreichen einer definierten Schwelle die On-board Kontrollalgorithmen autonom in den `SAFE` Mode übergehen. Im dann folgenden Verlauf der Drehrate erkennt man das systematische Verhalten der `SAFE` Mode Lageregelungsalgorithmen: Der Satellit erhöht seine Drehrate um die Achse mit dem größten Trägheitsmoment bis in eine Größenordnung von ungefähr 2°/sek (entspricht ~0,035 rad/sek) und richtet diese Achse so zur Sonne aus, dass die Solarpaneele zur Sonne orientiert sind. Der Orientierungswinkel β_{Sun} der Solarpaneele zur Sonne ist wieder für beide Szenarien identisch und in Abbildung 4.13 unten illustriert.

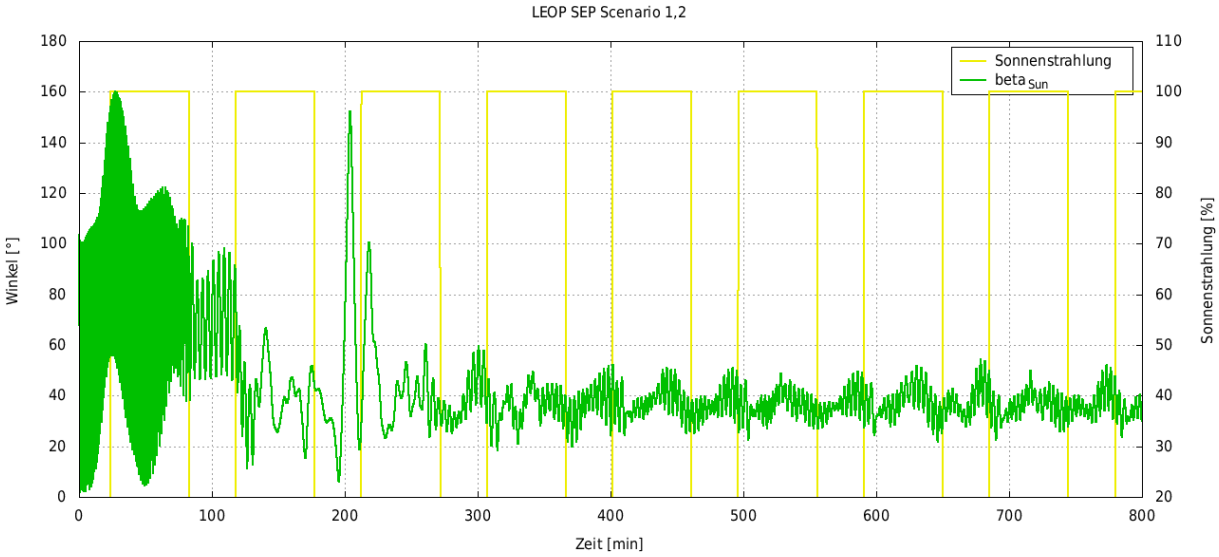


Abbildung 4.13: Orientierungswinkel β_{Sun} der Solarpaneele zur Sonne für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug

Im `DETUMBLE` Mode nach der Separation vom Orbittransferfahrzeug rotiert der Satellit anfangs mit einer hohen Drehrate, die im Verlauf der ersten 100 Minuten drastisch reduziert wird. Dies wird auch im Orientierungswinkel β_{Sun} der Solarpaneele zur Sonne im obigen Diagramm sichtbar. Beim Übergang in den `SAFE` Mode nach ungefähr 75 Minuten erkennt man während der zweiten Sonnenphase im Erdorbit, dass der Satellit beginnt, sich mit den Solarpaneelen zur Sonne auszurichten. Aufgrund der kleinen verbliebenen Drehraten in den drei Achsen (siehe Abbildung 4.12) und aufgrund der Drehratensteigerung in der Körperachse mit dem größten Trägheitsmoment gelingt die Sonnenausrichtung anfangs nur bedingt. Erst in der dritten Sonnenphase ist eine ansatzweise regelmäßige Ausrichtung zur Sonne möglich, die aufgrund der steigenden Drehrate in der Körperachse mit dem größten Trägheitsmoment auch einigermaßen stabil erhalten bleibt.

Die Energiebilanz ist für beide Szenarien für die ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug in Abbildung 4.14 gegeben. Auf den ersten Blick ergeben sich keine deutlichen Unterschiede. Dennoch ist der Entfaltungszeitpunkt der Solarpaneele deutlich im plötzlichen Anstieg der erzeugten Leistung der Solarzellen (im Diagramm in blau dargestellt) erkennbar. Im ersten Szenario erfolgt die Entfaltung nach ungefähr 75 Minuten, während im zweiten Szenario die Entfaltung bei Bodenkontakt per Telekommando nach ungefähr 125 Minuten erfolgt. Obwohl die Entfaltung der Solarpaneele im zweiten Szenario erst später erfolgt und damit die volle Leistung der Solarzellen erst später zur Verfügung steht, ist kaum ein Unterschied im Verlauf des Batterieladezustands zu erkennen. Die volle Batterieleistung steht fast zeitgleich zur Verfügung. Dies ist in Szenario 2 definitiv auf das sehr zeitnahe Kommando zur Entfaltung der Solarpaneele zurückzuführen, welches bereits im zweiten Erdorbit nach der Separation von der Missionskontrolle gesendet werden konnte. Sollte sich dieses Kommando verzögern, verzögert sich damit auch der Beginn der positiven Energiebilanz und damit verbunden die Aufladung der Batterie.

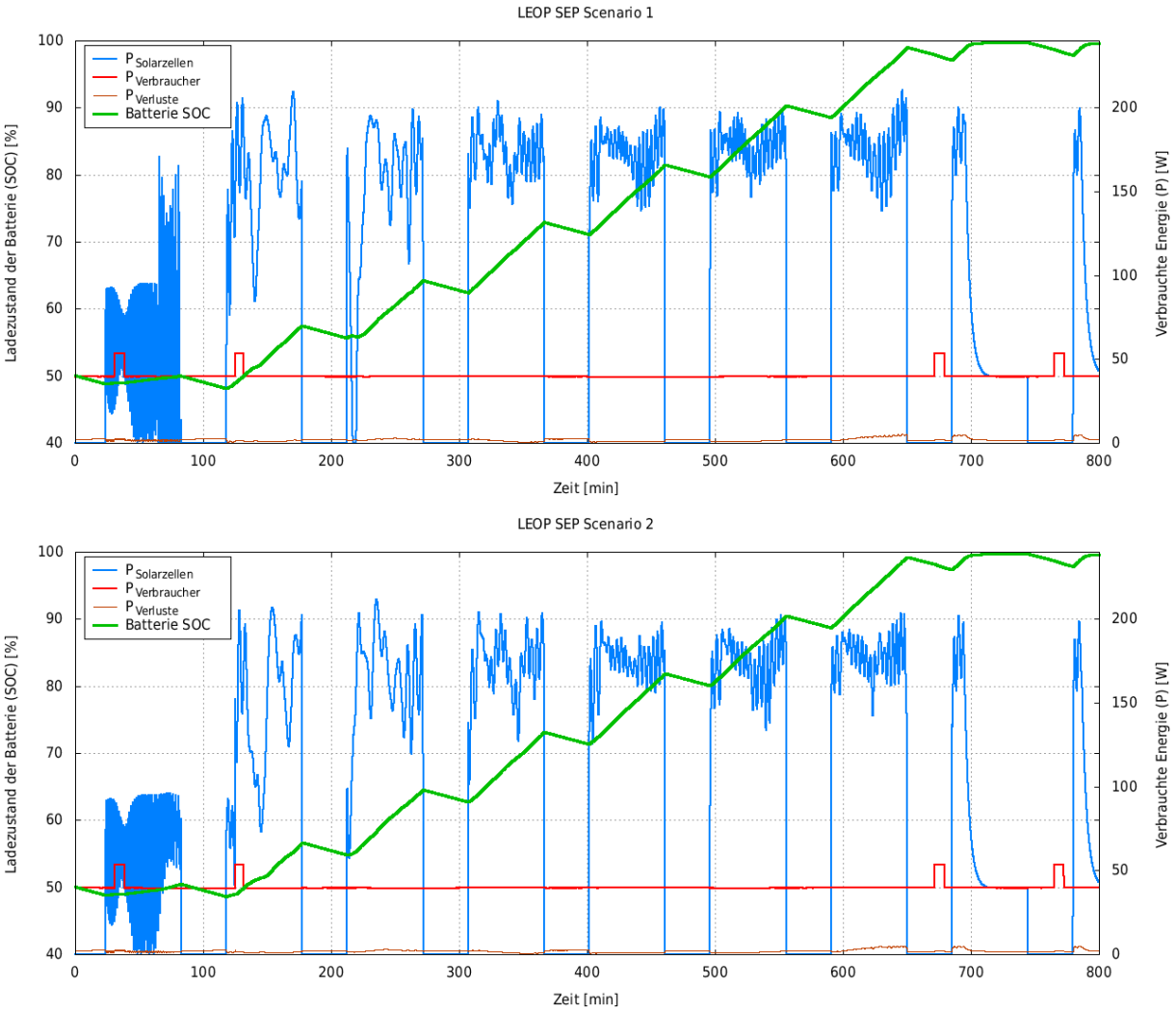


Abbildung 4.14: Energiebilanz für beide Szenarien im Verlauf der ersten 800 Minuten nach der Separation vom Orbittransferfahrzeug

Zusammenfassung

Die Simulationen von beiden Separationsszenarien zeigen aus energetischer Perspektive geringe aber charakteristische Unterschiede. Das Szenario 1 mit der autonomen Entfaltung der Solarpaneele beim Übergang in den `SAFE` Mode ist vorzuziehen, da keine Interaktion der Missionskontrolle durch ein Kommando zur Entfaltung der Solarpaneele erforderlich ist. Dadurch wird der Satellit schnellstmöglich in einen Zustand mit positiver Energiebilanz versetzt und ist in der Lage die beim Raketenstart nur halb geladene Batterie vollständig aufzuladen.

Ungünstiger ist die Situation, wenn das Entfalten der Solarpaneele von der Missionskontrolle aus kommandiert wird. In diesem Fall ist der Zustand des Satelliten bis zum Entfalten der Solarpaneele aufgrund der negativen Energiebilanz als kritisch zu betrachten und es muss nach der Separation vom Orbittransferfahrzeug ein schnellstmöglicher Funkkontakt zum Satelliten sichergestellt werden. Dies ist aber ein schwieriges Unterfangen, da die genauen Bahnparameter des Satelliten anfangs nicht bekannt sind und erst nach einiger Zeit bzw. einigen Erdumkreisungen zur Verfügung stehen. Somit ist das Auffinden des Satelliten im Erdorbit zur Kommunikation mit demselbigen zeitlich nicht sicher vorhersagbar.

4.3 Zusammenfassung der Ergebnisse

Als Voraussetzung für die Durchführung von Systemsimulationen mit den von EADS Astrium bereitgestellten Simulatorkomponenten sind vom Autor dieser Dissertation eine Reihe von Aufgaben bewältigt worden. Dazu gehören in erster Linie der Aufbau der Simulator-Infrastruktur mit der dazu erforderlichen Computer Hardware, der Erstellung eines projektbezogenen Datenbanksystems, die Schaffung eines Datenbankextraktionswerkzeugs zur Erzeugung stilkonformer XML-Konfigurationsdateien des Simulators, die Umsetzung eines CVS Repositoriums für die Quellcodeverwaltung und die Einrichtung des UML-Entwicklungswerkzeugs. Weiterhin ist mit dem von der ESA angebotenem Missionskontrollsystem SCOS-2000 die Beschaffung, Installation, Konfiguration und Inbetriebnahme eines anerkannten Systems zur Steuerung und Überwachung von Systemsimulationen realisiert worden. Ferner darf auch die durch das `MoonWiki` repräsentierte Wissensdatenbank nicht vernachlässigt werden, trägt sie doch entscheidend dazu bei, das erarbeitete Wissen im Kontext der Systemsimulation in einem Team mit hoher Fluktuationsrate zu erhalten.

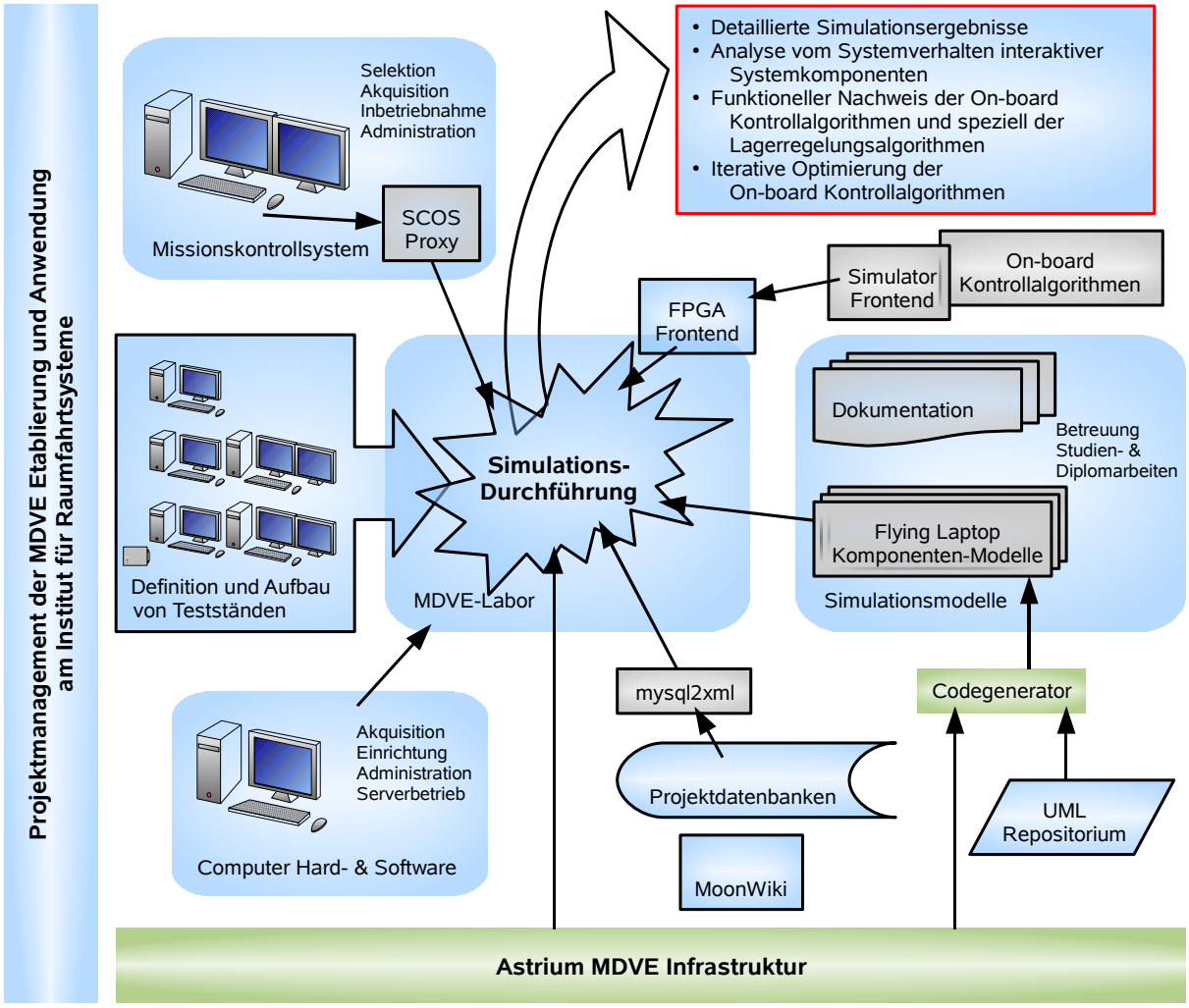


Abbildung 4.15: Graphische Zusammenstellung der Komponenten und Aufgaben zur Durchführung von Systemsimulationen; Die Leistungen des Autors sind in blau hervorgehoben

Zur Anwendung der Systemsimulation im Kleinsatellitenprojekt *Flying Laptop* sind alle relevanten Satellitenkomponenten im Simulator durch entsprechende, detaillierte Softwaremodelle abgebildet worden. Die Komponentenmodelle wurden unter Anleitung und Betreuung des Autors von Studenten im Rahmen von Studien- und Diplomarbeiten erstellt. Der sukzessiven Entwicklung der Komponentenmodelle ist durch den schrittweisen Aufbau von immer komplexeren Teststänten der Software-Verifikations-Einrichtung Rechnung getragen worden. Infolgedessen haben sich die Simulationsfähigkeiten von ersten Orbitsimulationen mit wenigen Komponentenmodellen, aber mit geschlossenem Simulationskreislauf, über die Simulation von Energie- und Thermalbilanzen unter Betriebsbedingungen bis hin zu vollständigen Systemsimulationen unter Einbindung der realen On-board Kontrollalgorithmen auf einem FPGA-Entwicklungsboard erweitert. Der *Flying Laptop* wird mit einem FPGA-basiertem On-board Computer System ausgestattet sein, welches durch ein hohes Maß an Parallelität gekennzeichnet ist. Gerade dieses charakteristische On-board Computer System mit den darauf betriebenen Kontrollalgorithmen führte zu einer großen Herausforderung bei der modellhaften Repräsentation derselbigen im Systemsimulator. In diesem Zusammenhang stellt die Einbindung der realen On-board Kontrollalgorithmen auf einem FPGA-Entwicklungsboard in den geschlossenen Simulationskreislauf eine erfolgreich realisierte technische Neuerung dar.

In Abbildung 4.15 ist eine graphische Darstellung der einzelnen Komponenten und erledigten Aufgaben zum Aufbau des Systemsimulators und zur Durchführung von Simulationen gegeben. Die vom Autor dieser Dissertation erbrachte Leistung ist in der Graphik blau hervorgehoben, wohingegen die beigesteuerten Leistungen aus Studien- und Diplomarbeiten grau koloriert sind. Die von EADS Astrium bereitgestellte Simulator-Infrastruktur ist in grün dargestellt.

Zum Zeitpunkt der Fertigstellung dieser Arbeit steht dem Projekt *Flying Laptop* durch die Einbindung der On-board Kontrollalgorithmen ein umfangreicher Teststand mit großem Potential zur Entwicklung und zum Funktionsnachweis der On-board Kontrollalgorithmen zur Verfügung. Dieser wurde bereits intensiv zum Testen der ersten Lageregelungsalgorithmen genutzt und kann auch zukünftig sukzessive mit Funktionserweiterungen der On-board Kontrollalgorithmen angewandt werden. Typische Missionsszenarien, wie die Transition zwischen zwei Betriebsmodi des Satelliten oder der automatische Übergang in den `SAFE` Mode bei einem Fehler, können jetzt simuliert und getestet werden. Die Ergebnisse dieser Simulationen dienen nicht nur dem reinen Funktionsnachweis der On-board Kontrollalgorithmen, sondern fließen direkt in die Optimierung der Kontrollalgorithmen zurück und führen so zu einem iterativem Verbesserungsprozess. Der Systemsimulator ist mit der Projektdatenbank als Teil seiner Infrastruktur ganz gezielt so implementiert worden, dass einzelne Testszenarien einfach und ohne wiederholten Konfigurationsaufwand reproduziert werden können. Somit unterstützt der Systemsimulator den sich in der Softwareentwicklung typischerweise iterativ wiederholenden Verifikationsprozess in einer optimalen Form.

5 Ausblick

Nachdem der Systemsimulator und das Missionskontrollsystem am Institut für Raumfahrtsysteme aufgebaut und auch durch intensive erste Simulationen und Tests der On-board Kontrollalgorithmen in Betrieb genommen worden ist, eröffnet sich nun nachfolgenden Arbeiten ein weites Betätigungsfeld für die zukünftigen Anwendungsaufgaben und Erweiterungen mit dem Systemsimulator MDVE.

An erster Stelle steht die Anwendung des Teststandes mit den auf dem FPGA-Entwicklungsboard eingebundenen On-board Kontrollalgorithmen für Simulationen und Tests mit den Lageregelungsalgorithmen des *Flying Laptop*. Sukzessive können so Funktionserweiterungen sowohl der Lageregelungsalgorithmen als auch anderer Aufgaben getestet und optimiert werden. Dies führt zu direkt verwertbaren und dringend erforderlichen Ergebnissen für das *Flying Laptop* Projekt. Konkreter Optimierungsbedarf besteht momentan für die Lageregelungsalgorithmen im `SAFE` Mode, die durch die Irritation des Sonnensensorsystems aufgrund der Erdalbedoreflektion noch zu keiner befriedigenden Lageausrichtung zur Sonne in der Lage sind. Iterative Tests mit optimierten `SAFE` Mode Lageregelungsalgorithmen können hier zu einer deutlichen Verbesserung führen und somit auch aus energetischer Sicht im `SAFE` Mode in der Solarpaneel-Konfiguration `C` einen stabilen und sicheren Zustand gewährleisten.

Weiterhin müssen häufige und der Sicherheit dienende Missionsszenarien, wie die Transition zwischen zwei Betriebsmodi des Satelliten oder der automatische Übergang in den `SAFE` Mode im Fehlerfall, intensiv unter einer Vielzahl von Randbedingungen auf ihre unbedingte Funktionalität hin überprüft werden. Möglichen Fehlerfälle sind im Systemdesign zu beachten, da im Grunde jede Systemkomponente unvorhergesehen ausfallen oder Fehlinformationen liefern kann. In jedem dieser Fälle müssen die On-board Kontrollalgorithmen in der Lage sein mit dieser Fehlfunktion umzugehen, sie zu beheben oder aber in einen sicheren Betriebszustand zu wechseln und anschließend der Missionskontrolle Informationen zur Fehleridentifikation bereit zu stellen. Mindestens das Prinzip zum Umgang mit unvorhergesehenen Fehlern oder unerwartetem Verhalten von Systemkomponenten muss intensiv getestet und verifiziert werden.

Zur Unterstützung der intensiven Simulationsdurchführungen kann auch das Missionskontrollsystem SCOS-2000 über eine Schnittstelle zur Ausführung von Skripten für Missionsszenarien mit hoher Interaktivität von Seiten des Operators genutzt werden. Dabei werden die zu sendenden Kommandos an den virtuellen Satellit bzw. an den Systemsimulator durch ein Skript automatisiert generiert und abgesendet. Eine ständige Interaktion und Beaufsichtigung durch den Operator ist dann nicht mehr zwingend erforderlich. Dies stellt bei sich wiederholenden Tests durch das nur einmalige Anfertigen eines Skriptes zum Einen eine starke Entlastung des Operators dar und gewährleistet im Weiteren die exakte Reproduktion des Testfalles. Gerade bei der Durchführung der Tests in Echtzeit ist dies ein wichtiges Merkmal, da diese durchaus länger als einen Arbeitstag in Anspruch nehmen können.

Der nächste wichtige Entwicklungsschritt ist der Aufbau eines Teststandes für den realen On-board Computer. In diesem Teststand werden die übertragenen Daten dann nicht mehr über das Simulator-Frontend gebündelt, sondern jede Ausgangsleitung des On-board Computers wird über eine eigene Kabelverbindung mit dem Systemsimulator über GMFE-Schnittstellenkarten verbunden. Dieser „Satellite-Testbed” genannte Teststand ist notwendig, um nicht nur

den Funktionsnachweis der On-board Kontrollalgorithmen zu erbringen, sondern auch das gesamte On-board Computer System mit seinen vier unabhängigen FPGA Boards und dem Zentralen Prozessierungsknoten zu testen. Weiterhin werden dabei auch die Hardware-Schnittstellen zu den Satellitenkomponenten inklusive der Algorithmen zum Ansteuern derselbigen von der FPGA Seite verifiziert.

Mit dem Satellite-Testbed können und müssen dann alle bereits mit der vorherigen Software-Verifikations-Einrichtung durchgeführten Missionsszenarien nochmals getestet werden. Erst dadurch können diejenigen Teile der Kontrollalgorithmen, die zum Ansteuern und zur Kommunikation mit den realen Hardwareschnittstellen dienen ebenfalls verifiziert werden.

Basierend auf dem Satellite-Testbed kann im Anschluss während der Integrationsphase des Satelliten das so genannte „Flatsat-Testbed“ erstellt werden. In diesem Teststand werden nacheinander einzelne reale Satellitenkomponenten an das On-board Computer System angeschlossen und die jeweilige entsprechende Komponente aus der Systemsimulation entfernt. Die nach jedem Integrationsschritt notwendigerweise durchgeführten Tests verifizieren die Kompatibilität zwischen dem On-board Computer System, den darauf betriebenen On-board Kontrollalgorithmen, der jeweiligen realen Satellitenkomponente und dem Satellitensystem als Ganzem. So entwickelt sich der einfache Teststand vom Satellite-Testbed mit vielen simulierten Komponenten sukzessive zu einem kompletten Satellitensystem im Flatsat-Testbed. Die bereits in den vorherigen Entwicklungsstufen durchgeführten Tests und Missionsszenarien können jetzt allerdings nur noch eingeschränkt verwendet werden, da die realen Sensoren, sofern sie im Teststand integriert sind, im Integrationslabor natürlich keine Weltraumbedingungen mehr messen können. Somit ist in diesem Test keine korrekte, wenn auch simulierte Weltraumumgebung mehr gegeben.

Bei der Anwendung der Systemsimulationsumgebung MDVE für die im Stuttgarter Kleinsatellitenprogramm weiterhin geplanten Kleinsatelliten *Perseus* und *Lunar Mission BW1* sind im Wesentlichen keine gravierenden Erweiterungen notwendig. Die geplante Mondmission *Lunar Mission BW1* erfordert gegebenenfalls eine Erweiterung bzw. Verbesserung des Orbitpropagators, um die Einflüsse der Gravitationskraft des Mondes detailliert zu berücksichtigen. Auch das für den *Flying Laptop* in einem niedrigen Erdorbit implementierte Thermalmodell muss den Gegebenheiten auf dem Weg zum Mond und dem wissenschaftlichen Aufenthalt dort vor Ort angepasst werden.

Für den ersten Kleinsatelliten des Instituts für Raumfahrtsysteme mit einem elektrischen Antriebssystem ist die Systemsimulationsumgebung bereits bestens ausgestattet. Die Aufprägung der Schubkraft beim Betrieb des Antriebssystems und die Berücksichtigung der Schubkraft in der Orbitvorhersage ist in einem detaillierten noch zu erstellenden Modell des Antriebssystems einfach möglich. Weiterhin müssen für die *Perseus* Satellitenkomponenten detaillierte Modelle für den Systemsimulator erstellt werden, sofern sie nicht vom *Flying Laptop* weiter verwendet werden können. Beim Aufbau einer eigenständigen Software-Verifikations-Einrichtung für den Kleinsatelliten *Perseus* sind natürlich auch die Daten in der Datenbank analog zum *Flying Laptop* bzw. entsprechend der *Perseus* Konfiguration einzupflegen. Das Gleiche gilt für die Mondmission *Lunar Mission BW1*. Grundsätzlich kann jedoch von vielen bereits für den *Flying Laptop* erledigten Aufgaben und Konfigurationen profitiert werden. Dies gilt für den Aufbau der Systemsimulationsumgebung, der Simulator-Infrastruktur, des Missionskontrollsystems, die detaillierten Satellitenkomponentenmodelle und die Erstellung von Missionsszenarien in vorkonfigurierten Testfällen.

6 Literaturverzeichnis

[Amini et al. '05] R. Amini, J. A. Larsen, R. Izadi-Zamanabadi, D. D. V. Bhanderi: *Design and Implementation of a Space Environment Simulation Toolbox for Small Satellites*. 56th International Astronautical Congress (IAC): Fukuoka, Japan, 17.-21. Oktober 2005.

[Aonix '06] *Ameos - The Next Generation UML Modeling Tool*. Aonix GmbH: 2006. Im September 2007: http://www.aonix.com/ameos.html

[Benz '07] R. Benz: *Projektmanagement und System Engineering*. Jährliche Vorlesungreihe am Institut für Raumfahrtsysteme der Universität Stuttgart: Stuttgart, 2007.

[Brandt '07] A. Brandt: *Development of Functional Models for Communication and Thermal Hardware and a Thermal Model for the Flying Laptop Microsatellite Simulation* (Diplomarbeit, IRS-07-S-02). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 21. Februar 2007.

[Curti et al. '07] *AVIO - Space Avionics Lab*. San Marco Project Research Centre (CRPSM), University of Rome: 2007. Im Januar 2008: http://crpsm.psm.uniroma1.it/prjavio.htm

[CVS '86] *CVS - The Concurrent Versions System*. D. Grune & CVS Team: 23. Juni 1986. Im September 2007: http://www.nongnu.org/cvs/

[Dicken et al. '00] H. Dicken, G. Hipper, P. Müssig-Trapp: *Datenbanken unter Linux*. MITP-Verlag: Bonn, 2000.

[Eclipse '01] *Eclipse - an open development platform*. Eclipse Foundation: November 2001. Im September 2007: http://www.eclipse.org/

[Eickhoff '08] J. Eickhoff: *Systemsimulation in der Satellitenentwicklung (ISSN 1866-7376)*. Manuskript zur Vorlesung am Institut für Raumfahrtsysteme der Universität Stuttgart: Stuttgart, 2008.

[Eickhoff et al. '03] J. Eickhoff, R. Hendricks, J. Flemmig: *Model-based Development and Verification Environment*. 54th International Astronautical Congress (IAC): Bremen, 29. September - 01. Oktober 2003.

[Eickhoff et al. '07] J. Eickhoff, A. Falke, H.-P. Röser: *Model-based design and verification - State of the art from Galileo constellation down to small university satellites*. Elsevier Limited: Acta Astronautica, Volume 61, Issue 1-6, Juni-August 2007.

[Enderle et al. '04] W. Enderle, C. Boyd, J. A. King: *Joint Australian Engineering (Micro) Satellite (JAESat) - A GNSS Technology Demonstration Mission*. The 2004 International Symposium on GNSS/GPS: Sydney, Australien, 06.-08. Dezember 2004.

[Falke '04] A. Falke: *Entwurf und Auslegung einer Energieversorgung für einen solarelektrischen lunaren Kleinsatelliten* (Diplomarbeit, IRS-04-S-13). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 28. Mai 2004.

[Falke et al. '04] A. Falke, M. Hartling, R. Laufer, H.-P. Röser: *Implementation of a simulation environment for software and hardware simulation of power management for a lunar small satellite mission*. 55th International Astronautical Congress: Vancouver, Kanada, 04.-08. Oktober 2004.

[Falke et al. '06] A. Falke, G. Grillmayer, S. Walz, F. Hesselbach, J. Eickhoff, H.-P. Röser: *LED in-flight calibration and model-based development of ACS algorithms for the university micro-satellite Flying Laptop*. Small Satellite Systems and Services - The 4S Symposium: Chia Laguna, Italien, 25.-29. September 2006.

[Fischer '08] J.-O. Fischer: *Anpassung und Erweiterung eines Thermalmodells des Kleinsatelliten Flying Laptop* (Diplomarbeit, IRS-08-S-10). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 8. April 2008.

[Fowler '03] M. Fowler: *UML konzentriert*. Addison-Wesley: München, Dezember 2003.

[FSF '85] *The Free Software Foundation (FSF)*. Free Software Foundation: 1995. Im September 2007: http://www.fsf.org/

[GCC '85] *The GNU Compiler Collection (GCC)*. GNU Project: 1985. Im September 2007: http://gcc.gnu.org/

[GDB '86] *The GNU Project debugger (GDB)*. GNU Project: 1986. Im September 2007: http://www.gnu.org/software/gdb/

[Gilmore '02] D. G. Gilmore: *Spacecraft Thermal Control Handbook, Teil 1 - Fundamental Technologies*. Aerospace Press: El Segundo, Kalifornien, Dezember 2002.

[GPL '91] *GNU General Public License (GPL)*. Free Software Foundation: Juni 1991. Im September 2007: http://www.gnu.org/licenses/#GPL

[Grillmayer et al. '05a] G. Grillmayer, A. Falke, H.-P. Röser: *Technology demonstration with the micro-satellite Flying Laptop*. 5th IAA Symposium on Small Satellites for Earth Observation: Berlin, 04.-08. April 2005.

[Grillmayer et al. '05b] G. Grillmayer, A. Falke, H.-P. Röser: *Technology Demonstration with the Micro-Satellite Flying Laptop*. Walter de Gruyter: Selected Proceedings of the 5th International Symposium of the International Academy of Astronautics, Volume 1, Issue 1, 04.-08. April 2005.

[GS & JSB '03] GS Battery USA & Japan Storage Battery: *Lithium Ion Cells for Space Applications* (Technische Dokumentation, Ausgabe 1.0). Joplin, USA, 07. Mai 2003.

[Kossev '08] I. Kossev: *Entwicklung von funktionalen Softwaremodellen der ACS-Aktuatoren und Integration eines Environment and Dynamics Propagators in die Simulationsumgebung des Kleinsatelliten Flying Laptop* (Diplomarbeit, IRS-07-S-03). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, Juni 2008.

[Kuwahara et al. '07] T. Kuwahara, F. Huber, A. Falke, M. Lengowski, S. Walz, G. Grillmayer, H.-P. Röser: *System design of the small satellite Flying Laptop, as the Technology Demonstrator of the FPGA-based On-board Computing System*. 58th International Astronautical Congress (IAC): Hyderabad, Indien, 24.-28. September 2007.

[Lachenmann '06] M. Lachenmann: *Utilization of Low-Cost-CMOS-/CCD-Cameras on a Small Satellite for Monitoring and Public Relations* (Diplomarbeit, IRS-06-S-29). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 23. August 2006.

[LaRoche '97] G. LaRoche: *Solargeneratoren für die Raumfahrt*. Vieweg: Braunschweig, Wiesbaden, 1997.

[Larson & Wertz '99] W. J. Larson, J. R. Wertz, et al.: *Space Mission Analysis and Design*. Kluwer Academic Publishers: Dordrecht, Niederlande, September 1999.

[Law & Kelton '00] A. M. Law and W. D. Kelton: *Simulation Modelling and Analysis*. Mcgraw-Hill Higher Education: New York u. a., 1. Januar 2000.

[MediaWiki '03] *MediaWiki - free wiki software package*. Wikimedia Foundation, Inc.: 08. Dezember 2003. Im September 2007: http://www.mediawiki.org/

[MoonWiki '06] *MoonWiki - The IRS Information Center*. Institut für Raumfahrtsysteme, Universität Stuttgart: 18. Januar 2006. Im September 2007: http://moonwiki.irs.uni-stuttgart.de/

[MySQL '96] *MySQL Database Server / Client*. MySQL AB: 1996. Im September 2007: http://www.mysql.com/

[Neumann '02] H. A. Neumann: *Objektorientierte Softwareentwicklung mit der Unified Modelling Language (UML)*. Hanser Fachbuchverlag: München, März 2002.

[OOo '02] *OpenOffice.org - the free office suite*. OpenOffice.org: 01. Mai 2002. Im September 2007: http://www.openoffice.org/

[Partimage '00] *Partimage - a Linux/UNIX utility which saves partitions in many formats to an image file*. F. Dupoux & F. Ladurelle: Juni 2000. Im September 2007: http://www.partimage.org/

[Röser et al. '05] H.-P. Röser, F. Huber, G. Grillmayer, M. Lengowski, S. Walz, A. Falke, T. Wegmann: *A small satellite of the University Stuttgart - A demonstrator for new techniques*. 31st International Symposium on Remote Sensing of Environment: Sankt Petersburg, Russland, 20.-24. Juni 2005.

[Ryan '07] *UML diagram types in hierarchically order*. Wikipedia: 30. März 2007. Im September 2007: http://en.wikipedia.org/wiki/Image:Uml_diagram.svg

[Saage '07] R. Saage: *Entwicklung von Softwaremodellen und Systemsimulation des Energieversorgungssytems des Kleinsatelliten Flying Laptop* (Studienarbeit, IRS-07-S-15). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 27. März 2007.

[ScopeSET '07] *OpenAmeos - Open Source UML Modeling Tool*. ScopeSET Technology Deutschland GmbH: 2007. Im September 2007: http://openameos.org/

[SysRescCd '03] *SystemRescueCd - a bootable CD-ROM Linux system for repairing and recovering*. F. Dupoux, P. Dorgueil, F. Ladurelle & I. Salinas: Juli 2003. Im September 2007: http://www.sysresccd.org/

[Tortora et al. '06] P. Tortora, F. Giulietti, A. Corbelli, V. Fabbri: *ALMASat Attitude Control Hardware-in-the-Loop Simulation*. 57th International Astronautical Congress (IAC): Valencia, Spanien, 02.-06. Oktober 2006.

[UML '97] *UML® Resource Page*. Object Management Group: November 1997. Im September 2007: http://www.uml.org/

[W3C '94] *The World Wide Web Consortium*. World Wide Web Consortium: Oktober 1994. Im Oktober 2007: http://www.w3.org/

[Wikipedia '01] *Wikipedia - The Free Encyclopedia*. Wikimedia Foundation, Inc.: 15. Januar 2001. Im September 2007: http://www.wikipedia.org/

[Wikipedia '08a] *Datenbanksystem*. Wikipedia: 09. April 2008. Im April 2008: http://de.wikipedia.org/w/index.php?title=Datenbanksystem&oldid=44690845

[Wikipedia '08b] *Nyquist-Shannon-Abtasttheorem*. Wikipedia: 23. März 2008. Im April 2008: http://de.wikipedia.org/w/index.php?title=Nyquist-Shannon-Abtasttheorem&oldid=44060566

[Wolter '06] V. Wolter: *Interfacing the attitude control devices to the on board computer of the micro satellite Flying Laptop* (Diplomarbeit, IRS-06-S-28). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 15. März 2006.

[Wyszkowski '05] B. WYSZKOWSKI: *Erstellung von Softwaremodellen für die MDVE-Systemsimulation der Sensorik des Lageregelungssystems für den Kleinsatelliten Flying Laptop* (Diplomarbeit, IRS-05-S-39). Stuttgart: Institut für Raumfahrtsysteme, Universität Stuttgart, 20. Dezember 2005.

[XML '98] *Extensible Markup Language (XML)*. World Wide Web Consortium: 10. Februar 1998. Im Oktober 2007: http://www.w3.org/XML/

[Zeigler '76] B. P. ZEIGLER: *Theory of Modelling and Simulation*. John Wiley & Sons Inc.: New York u. a., Juli 1976.

[Zheng, Low '06] *Virtual Instrument for a Micro-satellite Power Supply System*. Satellite Engineering Centre, Nanyang Technological University (NTU): 2006. Im Januar 2008: http://digital.ni.com/worldwide/singapore.nsf/web/all/2B2D1AF53F882E97862572460042AF1F

Danksagung

Selbst zu Beginn des 21. Jahrhunderts stellt der Entwurf und der Betrieb eines Satellitenprogramms eine sehr große technische, finanzielle und organisatorische Herausforderung dar. Nicht jeder Student bzw. Diplom Ingenieur erhält die Möglichkeit, an einem solch ehrgeizigen Programm mitzuarbeiten und Verantwortung zu übernehmen. An dieser Stelle möchte ich mich daher bei Herrn Professor Dr. Hans-Peter Röser herzlich für das in mich gesetzte Vertrauen, die Betreuung meiner Promotion und die Bereitstellung der räumlichen und technischen Möglichkeiten bedanken. Durch die Implementation einer Systemsimulationsumgebung konnte ich bei der Entwicklung des Kleinsatelliten *Flying Laptop* tatkräftig mitwirken.

Für die intensive Betreuung meiner Arbeit, die detaillierten Erläuterungen und vielen fruchtbaren Diskussionen und Anregungen gilt mein besonderer Dank Herrn Dr. Jens Eickhoff, der von Seiten des Industriepartners EADS Astrium GmbH meine fachliche Betreuung übernahm. Seine stete Hilfsbereitschaft und sein umfassendes Wissen die Systemsimulation betreffend weiß ich sehr zu schätzen.

In diesem Zusammenhang möchte ich auch EADS Astrium in Friedrichshafen danken, die durch die kooperative Bereitstellung von wesentlichen Simulatorkomponenten die Bearbeitung eines solch umfangreichen Themas innerhalb eines Kleinsatellitenprojektes erst ermöglicht haben. Für die kontinuierliche Unterstützung während der Aufbau- und Nutzungsphase des Simulators möchte ich mich namentlich bei den Mitarbeitern Marco Kahlmeier, Andrea Marongiu und Björn Kircher bedanken.

Des Weiteren gilt mein Dank allen Aktivisten im Stuttgarter Kleinsatellitenprogramm für die fruchtbare Zusammenarbeit, die kritischen Betrachtungen und die angenehm lebhafte Atmosphäre, in der ich mich sehr wohl gefühlt habe. Namentlich bedanke ich mich bei den ehemaligen Studenten Boris Wyszkowski, Alexander Brandt, Ivan Kossev, Rainer Saage, Sascha Tietz, Michael Fritz, Jens Fischer und Claas Ziemke die ihre Studien- bzw. Diplomarbeit oder Hiwi-Tätigkeit im Rahmen der MDVE Systemsimulation durchgeführt haben. Sie trugen einen spürbaren und hilfreichen Beitrag zum Gelingen dieses Projekts bei.

Insbesondere möchte ich an dieser offiziellen Stelle meinen Eltern Danke sagen, die mir eigenständigen Freiraum für jedes meiner Ziele geschaffen und mich stets in jeder Hinsicht unterstützt haben.

Mein herzlichster Dank gilt meiner Frau Birgit, die nicht nur meine zuweilen langen und späten Arbeitsstunden entschuldigte, sondern mir in den richtigen Augenblicken vergegenwärtigte, dass neben der fachlichen Berufung weiteres Lebenswertes existiert.

Publikationsliste des Autors

Referierte Journale

A. Brandt, I. Kossev, A. Falke, J. Eickhoff, H.-P. Röser: *Preliminary system simulation environment of the university micro-satellite Flying Laptop*. Selected Proceedings of the 6th International Symposium of the International Academy of Astronautics, 23.-26. April 2007.

J. Eickhoff, A. Falke, H.-P. Röser: *Model-based design and verification - State of the art from Galileo constellation down to small university satellites*. Acta Astronautica Volume 61, Issues 1-6, Juni-August 2007, Seiten 383-390. Bringing Space Closer to People, Selected Proceedings of the 57th IAF Congress, Spanien, Valencia, 2.-6. Oktober 2006.

G. Grillmayer, A. Falke, H.-P. Röser: *Technology Demonstration with the Micro- Satellite Flying Laptop*. Selected Proceedings of the 5th International Symposium of the International Academy of Astronautics, Seiten 419-427. Deutschland, Berlin: 4.-8. April 2005.

Konferenzbeiträge

T. Kuwahara, F. Huber, A. Falke, M. Lengowski, S. Walz, G. Grillmayer, H.-P. Röser: *System Design of the Small Satellite Flying Laptop, as the Technology Demonstrator of the FPGA-based On-board Computer System (IAC-07-B4.6.08)*. Indien, Hyderabad: 55th International Astronautical Congress, 24.-28. September 2007.

A. Brandt, I. Kossev, A. Falke, J. Eickhoff, H.-P. Röser: *Preliminary system simulation environment of the university micro-satellite Flying Laptop (IAA-B6-0603)*. Deutschland, Berlin: 6th IAA Symposium on Small Satellites for Earth Observation, 23.-26. April 2007.

J. Eickhoff, A. Falke, H.-P. Röser: *Model-based Design and Verification - State of the Art from Galileo Constellation down to Small University Satellites (IAC-06-D1.3.02)*. Spanien, Valencia: 57th International Astronautical Congress, 2.-6. Oktober 2006.

A. Falke, G. Grillmayer, S. Walz, F. Hesselbach, J. Eickhoff, H.-P. Röser: *LED in-flight calibration and model-based development of ACS algorithms for the university micro-satellite Flying Laptop.* Italien, Chia Laguna: Small Satellite Systems and Services - The 4S Symposium, 25.-29. September 2006.

H.-P. Röser, F. Huber, G. Grillmayer, M. Lengowski, S. Walz, A. Falke, T. Wegmann: *A small satellite of the University Stuttgart a demonstrator for new techniques.* Russland, Sankt Petersburg: 31st International Symposium on Remote Sensing of Environment, 20.-24. Juni 2005.

G. Grillmayer, A. Falke, H.-P. Röser: *Technology demonstration with the micro-satellite Flying Laptop (IAA-B5-1402)*. Deutschland, Berlin: 5th IAA Symposium on Small Satellites for Earth Observation, 4.-8. April 2005.

H.-P. Röser, M. v. Schoenermark, F. Huber, G. Grillmayer, M. Lengowski, S. Walz, A. Falke, T. Wegmann: *Earth Observation with the Small Satellite "Flying Laptop" of the University of Stuttgart.* Frankreich, Straßburg: 9th ISU Annual International Symposium, 30. November - 3. Dezember 2004.

A. Falke, M. Hartling, R. Laufer, H.-P. Röser: *Implementation of a simulation environment for software and hardware simulation of power management for a lunar small satellite mission (IAC-04-IAA.4.11.6.05)*. Kanada, Vancouver: 55th International Astronautical Congress, 4.-8. Oktober 2004.

A. Falke: *Thermal investigation of a super-pressurized balloon in martian environment (DGLR-2004-048NW)*. Deutschland, Dresden: DGLR Jahrestagung 2004, 21. September 2004.

LED IN-FLIGHT CALIBRATION AND MODEL-BASED DEVELOPMENT OF ACS ALGORITHMS FOR THE UNIVERSITY MICRO-SATELLITE FLYING LAPTOP

A. Falke[(1)], G. Grillmayer[(1)], S. Walz[(1)], F. Hesselbach[(1)], J. Eickhoff[(2)], H.-P. Roeser[(1)]

[(1)] Universität Stuttgart, Pfaffenwaldring 31, 70569 Stuttgart, Germany, Email: roeser@irs.uni-stuttgart.de
[(2)] EADS Astrium GmbH, 88039 Friedrichshafen, Germany, Email: jens.eickhoff@astrium.eads.net:

ABSTRACT

The goal of this paper is to describe two major points in the design and development of the micro-satellite Flying Laptop, currently built at the Institute of Space Systems of the Universität Stuttgart. The first point is the in-flight calibration system using LEDs for the MICS (multi-spectral imaging camera system) instrument. This instrument will measure the bi-directional distribution function (BRDF) in the three spectral channels: green, red and near infrared. For the scientific evaluation of the data, an in-flight calibration unit is mandatory. The second point is the development approach of the attitude control algorithms. In order to decrease cost and start development prior to flight hardware availability, a software simulation based approach is realized. Hardware characteristics of each specific equipment unit are modelled in this extensive simulation environment allowing more detailed simulations. Later on the real flight hardware will be integrated in the loop successively replacing the software models.

1. INTRODUCTION

The Institute of Space Systems (IRS) at the Universität Stuttgart is currently developing several small satellite missions with the long term goal to launch a small satellite to the moon. All missions will be built and operated by the Institute of Space Systems. In this way direct access and control of the satellite and data is ensured. The first mission is called Flying Laptop [1] and has a cubical shape with an edge length of 60 cm x 70 cm x 80 cm and a mass of about 100 kg.

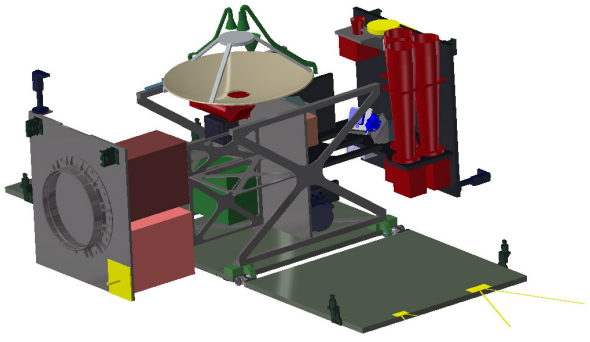


Figure 1. Flying Laptop

The satellite is 3-axis stabilized and a piggyback launch with the Polar Satellite Launch Vehicle (PSLV) is pursued at the beginning of 2008.

2. SCIENTIFIC OBJECTIVES

The second mission objective of the Flying Laptop, beside technology demonstration, is Earth observation. As described in [2] the main field of interest is the measurement of the bi-directional reflectance distribution function (BRDF). The reflectance of most surfaces shows anisotropic behaviour, which means, that the reflectance varies with solar and viewing positions. This effect is mathematically described by the BRDF which characterizes how for a scene with given properties, the reflectance observed varies with the angles of observation and illumination In order to measure this function the satellite is equipped with the MICS (multi-spectral imaging camera system) payload consisting of three identical cameras with different interference filters for the green, red and near infrared spectral wavelength. The key characteristics of the instrument are listed in Table 1.

Table 1. Key characteristics for the MICS instrument

Spectral bands (half width): Green Red Near Infrared	 530 nm – 580 nm 620 nm – 670 nm 820 nm – 870 nm
Ground Sample Distance	25 m
Swath width	25 km
Digitalization	12 bit
Power consumption	~ 5 W
Size	100 x 90 x 400 [mm]
Mass	~ 4 kg

The measurements of the BRDF will be done in the target pointing mode, where the satellite is focused on the target site during the whole passage.
In order to accomplish reliable scientific measurements a calibration of the instrument is mandatory, not only on ground, but also in space.

3. CALIBRATION UNIT

3.1. Current Situation

While technology demonstration is currently the dominating mission objective for small satellites and especially for University missions, scientific tasks are often neglected. One of the reasons is that reliable data

needs to be calibrated in flight. Unfortunately the calibration unit commonly increases the complexity, mass, size, power consumption and cost of an instrument significantly. Somehow the light of a calibration source has to be injected into the optical path, which requires a critical tilting mechanism. For example a diffuser is needed for sun calibration or a tilting mirror to inject the light of an on-board calibration lamp. Therefore most of the systems are only calibrated on ground and scientific measurements become difficult.
With the growing market of LED products in every day life, they are more and more used for calibrating purposes. Terrestrial calibration light applications are growing rapidly. Investigations for implementation of LEDs for space applications have been accomplished [3] and, although quite rare, some instruments already use LEDs as in-flight calibration source in space, for example the Indian Remote Satellite IRS-1C, which has a LED light source implemented for radiometric calibration of its PAN camera, or GOME on ERS-2, which uses LEDs for flatfield calibration.

3.2. Light Emitting Diodes

LEDs are special types of diodes, with the typical p-n junction. Current can flow only from the p-side to the n-side, but not inverse. The band gab energy of the chosen semiconductor material determines the emitted light. Therefore a variety of different spectral LEDs is available, through the use of different materials. Meanwhile the range of LEDs leeds from the ultraviolet spectral region to the infrared. Also different types of diodes exist, e.g. bare chip devices, SMD (surface mounted device) or the most common devices with epoxy lens packages. Comparing LEDs with traditional calibration lamps, the advantages are:

- Small size
- Low power consumption, mass and costs
- Different spectral availabilities
- Characteristic spectral emittance
- Linear intensity dependency on current

The disadvantages of LEDs are:

- Temperature dependent
- Reliability
- Heritage

One of the main advantages is the characteristic spectral emittance of LEDs with a typical full width at half maximum of about 40-80 nm and the availability of diodes with centre wavelengths over the whole visible spectrum. The main point, why they have not been used more often is the temperature dependency and the resulting radiance variation. This intensity variation is described by eq. 1 after [4]:

$$I = I\big|_{0K} \exp\left(-T / T_1\right) \quad (1)$$

where T_1 is the so called characteristic temperature, which describes the temperature dependency of the LED. This characteristic temperature differs for all different types of diodes.
Further the flight heritage and reliability data is poor compared to classical calibration lamps. Anyhow the key point for successful use of LEDs is the temperature stabilization.

3.3. Calibration Unit Concept

For the Flying Laptop a calibration unit concept was designed, in order to verify the following points:

- Pixel-to-pixel shift (flatfield calibration)
- Spectral shift of interference filters
- Radiometric performance

The sensibility and behavior of pixels changes within a CCD device due to fabrication. This pixel-to pixel shift can also change unequally through aging and the harsh space environment. In order to compensate this effect a flat field calibration becomes necessary in flight. Interference filters are of major interest for earth observation missions, because the filter curve edges are steep and they are close to a desired rectangular characteristic. Unfortunately, they tend to shift due to radiation effects. Hence, their behavior has to be monitored in space. With the narrow spectral bands of the LEDs, originally located at the filter middle wavelength changes in the filter characteristics can be recorded. The most difficult calibration with LEDs will be an absolute radiometric calibration. With two different current levels, the linearity of the CCD can be measured, but only the in flight experiment and comparison with other calibration methods, like vicarious calibration, will show, if the stability of the LEDs will be good enough to perform absolute reliable radiometric calibration.

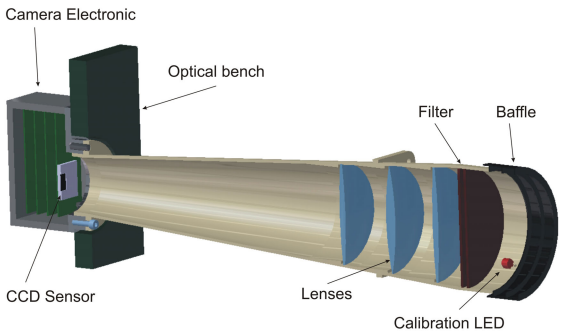


Figure 2: Schematic sectional drawing of MICS.

Following the design philosophy to keep all systems as simple as possible and to avoid moving parts the

calibration unit was designed under these aspects. The unit will be placed in front of the filters at the top end of the optics as shown in Figure 2

For each camera a bare chip device with a middle wave length equal to the filter middle wavelength was selected. The diodes are encapsulated on a TO18 socket together with a resistant silicone temperature sensor. In this configuration, shown in Figure 3, temperature measurement close to the emitting diode is ensured. Bare chip LEDs are used, in order to avoid contamination of the optic through outgasing effects of the epoxy lenses. A passive thermal concept was selected for temperature stabilization. The LED will be mounted in an anodized aluminum fitting for short term heat storage. Also, the diodes will be operated in a pulsed way, in order to reduce the heat dissipation and to increase the life time. For the electrical design a constant electrical current connection was chosen for operation. In this way, voltage fluctuations through temperature changes are avoided. Two current levels (15 mA and 20 mA) were chosen for operation and change of intensity levels. For redundancy reasons each camera is equipped with two LED units.

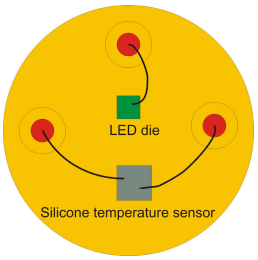


Figure 3: TO18 socket with LED and temperature sensor.

The fixed position of the LEDs in front of the filters within the aperture has no significant influence on the image quality. Since the focus of the optic is infinity, the units only reduce the aperture area by 1 %, which is the equivalent to a 2 % loss of incoming photons (signal). This decrease was verified through a lower signal level in the laboratory and is tolerable for the Flying Laptop mission. Further it was examined, if the units produce diffraction patterns, but no effects were detected.

4. MEASUREMENTS

As explained in section 3.2, light emitting diodes are strongly dependent on the temperature. In order to better understand this behaviour and to better evaluate the performance of a calibration unit with LEDs that can be expected, this dependency was examined with the chosen devices. The results will give a hint on the expected accuracy of a possible absolute radiometric calibration.

4.1. Laboratory Assembly

Figure 4 shows the laboratory assembly for the measurement of the LED optical changes with different temperatures. For heat transportation and temperature regulation, one diode is mounted on a aluminium housing connected to peltier elements. The diode is operated in continuous mode, with a constant current of 20 mA. The peltier elements regulate the assembly to a given temperature, which is reached after a few minutes. The collimator has an aperture of 1.5 mm and focuses the incoming light on an optical waveguide for the spectrometer input. The spectrometer is a fibre optical spectrometer USB 2000 from Ocean Optics with an spectral resolution of < 3 nm.

The on chip silicone temperature sensor gives the reference temperature for all measurements.

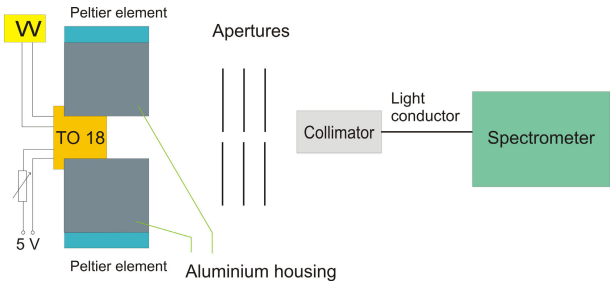


Figure 4: Laboratory assembly for temperature characterisitic.

4.2. Results

First the linearity of the LED for fixed currents was checked and showed the expected results. This behaviour is important for the radiometric linearity calibration.

Looking at the middle wavelength of the LED, a linear shift over temperature occurs. This effect is due to the changes in the kinetic energy of the electrons, which has an influence on the wavelength defining bandgap energy. As shown in Figure 5 the variation is linear and about 1 nm per 10 degrees Celsius. For the verification of the interference filter characteristics, this effect must be compensated.

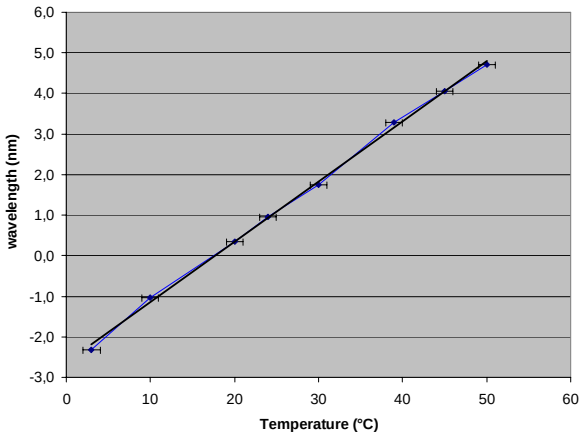


Figure 5: Middle wavelength displacement over temperature.

Figure 6 shows the normalized signal over a temperature range from zero to fifty degrees Celsius. The signal is normalized at 20 degree Celsius and is displayed in logarithmical scale. The intensity decrease has exponential characteristic which is consistent with the theoretical dependency described in eq.1. However, the characteristic temperature of the measured LED is unknown. From the results one can conclude, that it must be high, because of the quite stable behaviour. Hence the LED seems to be suitable for the foreseen task. Anyhow the changes in intensity level are problematic for calibration tasks, but with the temperature sensor close to the LED this effect as well as the shift of the middle wavelength can be quantified and therefore taken into account.

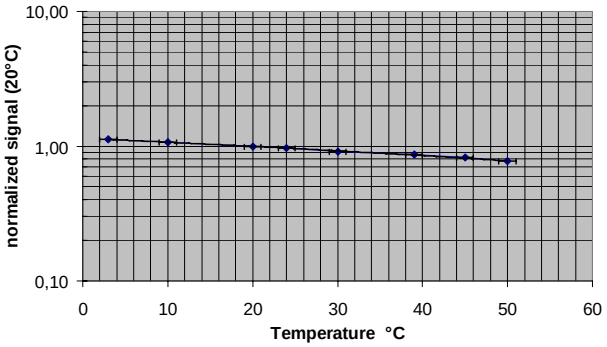


Figure 6: Normalized signal over temperature.

5. ATTITUDE CONTROL SYSTEM DEVELOPMENT

A micro-satellite typically has limited power and computational resources. In order to achieve the required agility, accuracy of the navigation & attitude control system and autonomy along with the parallel processing of the payload images in real-time, for the *Flying Laptop* the decision was made to use an FPGA based on-board computer (OBC).

5.1. ACS Hardware

The satellite motion is monitored by five different types of sensors: two three-axis magnetometers, two coarse sun sensors, four fiber-optic rate sensors, one autonomous star tracker with two camera heads and three GPS receivers. The actuators that rotate the satellite to the desired attitude are four reaction wheels and three magnetic torquers. All sensors and actuators are connected to the FPGA in a star like configuration. The attitude control system (ACS) hardware devices are shown in Figure 7 and Figure 9.

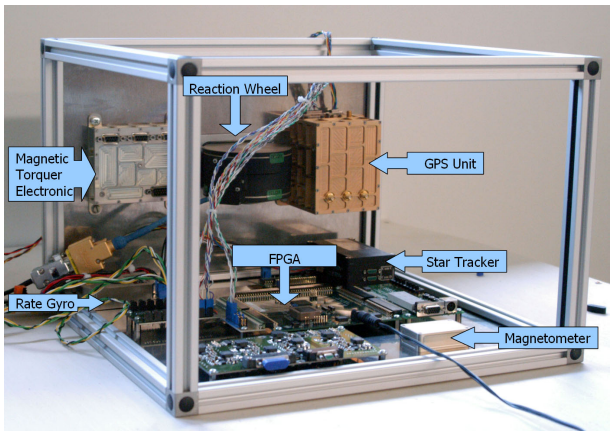


Figure 7: ACS test assembly

The ZARM-Technik AMR-magnetometer is a micro controller based 3-axis magnetometer with digital output. Two magnetometers will be installed on the *Flying Laptop*. The sun sensors system consists of two three-axis Coarse Sun Sensors from Sun Space mounted at diagonal corners of the satellite. The sensors voltages are digitized and sent to the OBC via an I^2C bus. Four C-FORS fiber optic gyros from Litef measure the angular rate. They are arranged in a tetrahedron configuration, to allow for a single failure. The sensors have a drift rate of 3°/h. All four rate sensors are connected to the on-board computer by a common IBIS bus. The micro-Advanced Stellar Compass from the Technical University of Denmark is used. The system consists of two camera head units connected to a data processing unit. The star tracker has to provide an attitude knowledge of 7 arcseconds for the *Flying Laptop* mission. The GENIUS (GPS Enhanced Navigation Instrument for the Universität Stuttgart micro-satellite) system is developed as an experiment for accurate determination of the spacecraft attitude using GPS. It consists of three COTS Phoenix GPS boards. Each of the receivers is connected to separate GPS antenna via a low noise amplifier. The antennas are placed at three corners of the middle solar panel in an L-

like arrangement. Furthermore the receivers are synchronized via an ultra-stable 10 MHz oscillator. Four reaction wheels RSI 01-5/28 from Teldix are running in a single hot redundant tetrahedron configuration. Each of the wheels has an angular momentum capacity of 0.12 Nms and a reaction torque of 5 mNm over the range of ±3000 rpm. Three ZARM-Technik magnetic torquers (torque rods) with a linear dipole moment of 6 Am² are utilized. The torquers are connected to a power box that includes two I^2C buses for connection to the OBC. The whole system is single redundant.

5.2. ACS Algorithms

Figure 10 shows a global overview of the ACS algorithms. Each block shown represents a subfunction with input and outputs. On the left side the inputs from the 5 sensors are shown representing the interface variables to the lower hardware dependent levels. The Processing section merges, converts, extrapolates the sensor variables to the current time and also contains the Kalman filters. In the Navigation part the reference attitude and angular velocity are calculated depending on the pointing mode selected. The Control section contains the safe mode, detumbling, desaturation, nullspace, inertial, nadir and target pointing controller [5]. For the last three an error quarternion feedback controller is used. The torque output from the controllers is scaled and processed in the Command section and sent to the wheels and torquerods shown at the right side.

5.3. ACS Algorithm Development Process

Figure 8 shows the development cycle for the ACS navigation and control algorithms. After initial theoretical analysis the algorithms are implemented in Matlab/Simulink and tested extensively for performance. All functions which are going to be implemented in the on-board computer are written as an embedded m-file. In the next step the algorithms have to be converted to Handel-C, which is a high level language compiler generating the binary netlist for the FPGA. Control algorithm and filters contain a large amount of calculations which require a large amount of gates in the FPGA. In order to limit the space needed on the FPGA almost all variables are converted to fixed point arithmetic and comparing the solution to the original double precision algorithm. The comparison is still performed in the Matlab/Simulink environment and is identical to the approach used for digital signal processors. After compilation the netlist is loaded to a prototype FPGA board (Figure 7) for initial testing. For final verification of the algorithms a real time simulation environment is needed. A model-based system-simulation environment represented by the MDVE (Model-based Development and Verification Environment), which is explained in the next section, is used.

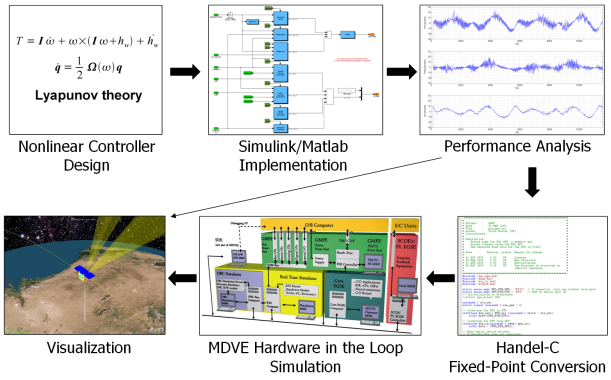


Figure 8: ACS development process

6. A MODEL-BASED DEVELOPMENT AND VERIFICATION APPROACH

In the past few years Model-based Development and Verification has become a very important role in the satellite design process of industry lead space projects. Since 2001 EADS Astrium GmbH for example has implemented a model-based system simulation infrastructure to support spacecraft development, on-board software verification and spacecraft design validation in order to accomplish the needs of a newly established engineering process [6]. This simulation infrastructure is called "Model-based Development and Verification Environment" (MDVE).

The simulation environment provides a tool infrastructure allowing S/C models ranging from early pure virtual simulations via hybrid testbenches up to full FlatSat configurations to be setup. This largely minimizes cost for hardware models, however in parallel provides risk mitigation through stepwise verification of onboard SW, onboard HW, flight procedures etc. first on simulation testbenches and later by hardware-in-the-loop configurations.

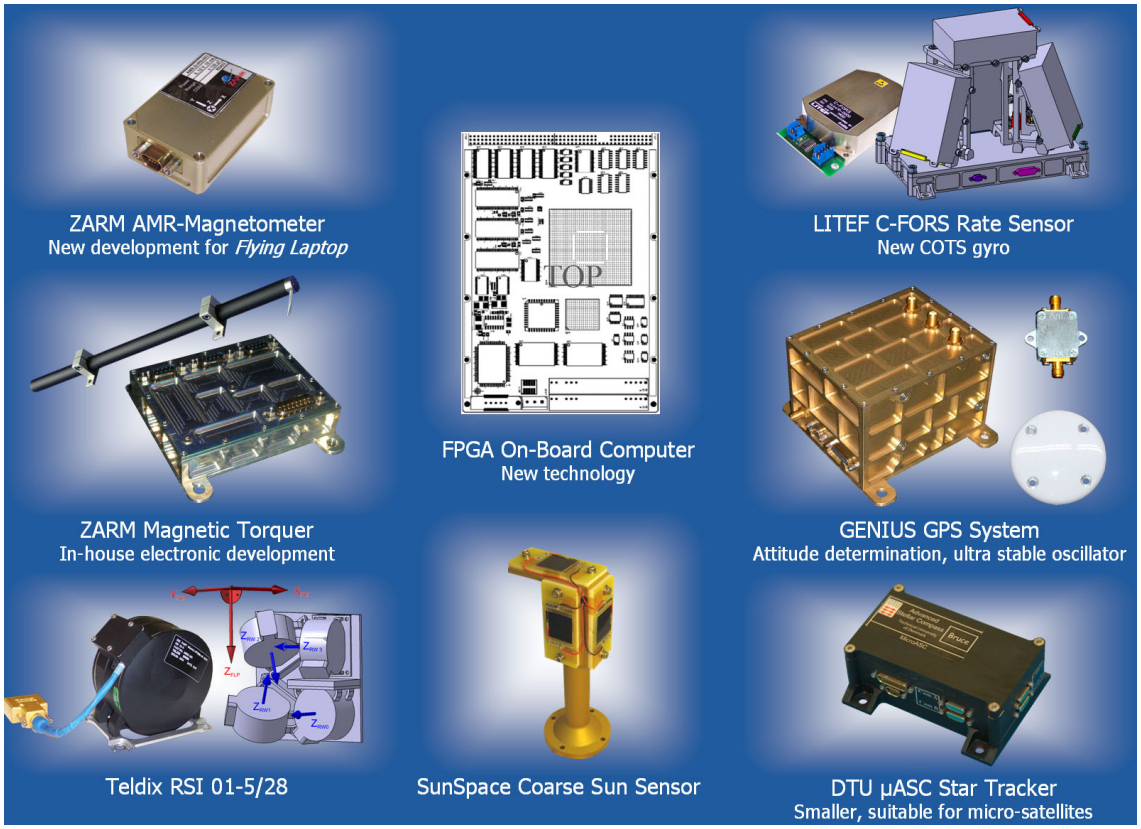


Figure 9: ACS sensors and actuators

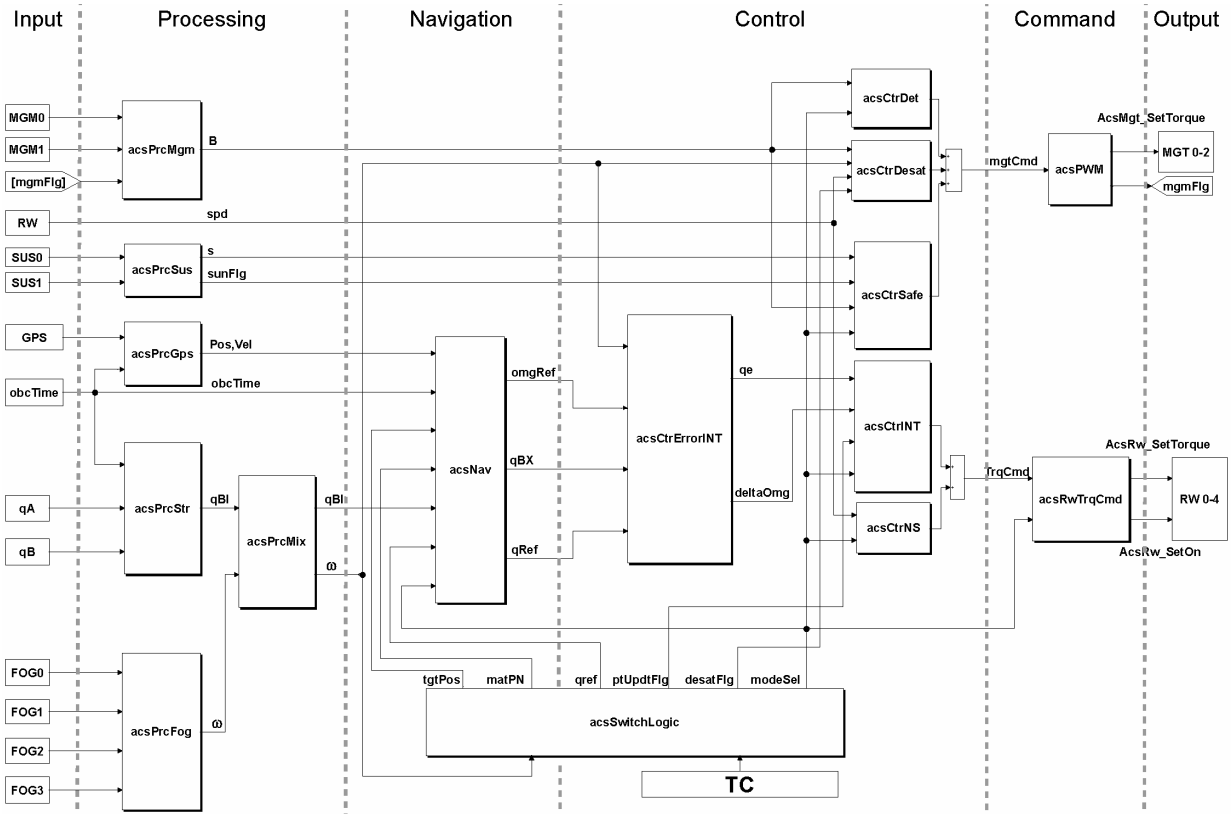


Figure 10: ACS subfunction map

6.1. MDVE Technology and Configurations

The MDVE will be used in various project phases for a number of design, development and verification tasks in different working environments. The most important MDVE standard configurations are:

- Development SVF (DEV-SVF)
- Software Verification Facility (SVF)
- Real-Time Testbed (RTB)
- Extended Real-Time Testbed (FlatSat)
- Spacecraft simulator for the mission control center.

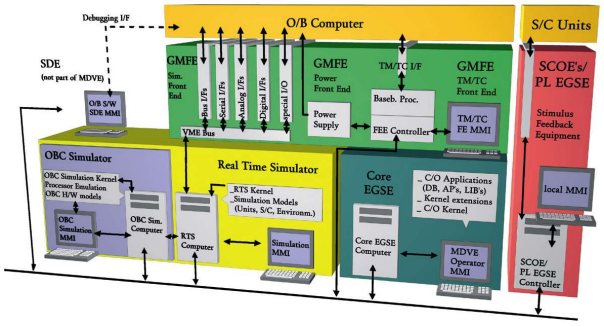


Figure 11. Building blocks and interfaces of the MDVE construction kit [7]

The core element of the MDVE is an On-board Computer Simulator (OBC-Simulator) and a Real-Time Simulator (RTS) modeling the remaining equipment units of the spacecraft (S/C) plus space environment conditions (refer to Figure 11). In exclusively simulation setups these simulators (synchronized to each other) are commanded via a control console – in most cases a Core EGSE.

The functional behavior of the satellite system and all interactions of the on-board equipment is modeled on the basis of the system and equipment specifications. First, the Software Verification Facility as illustrated in Figure 12, is set up using an adequate on-board computer simulator and, in respect of its electrical / functional architecture, clearly defined models of the satellite equipment. From now on, the SVF is used throughout the satellite life cycle to support debugging and implementation of new releases to the flight software as well as for the development of on-board control procedures.

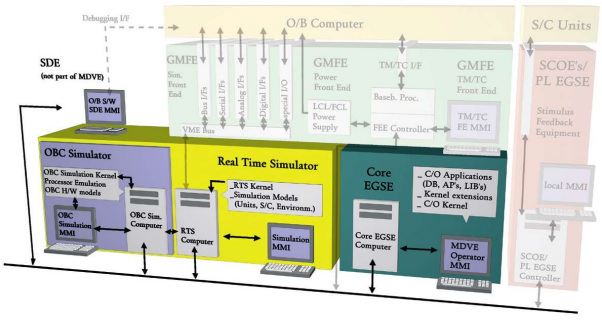


Figure 12. Software Verification Facility (SVF) [7]

In a next step the so-called Real-Time Testbed is set up, usually including the onboard computer as first item into the hardware-in-the-loop testbench. The OBC, which might be a bread board or engineering model, replaces the OBC-Simulator and is connected to the Real-Time Simulator (RTS) – now really running under hard real-time conditions to serve the OBC. Hardware / software integration and compatibility verification of the flight software under real hardware conditions is now possible.

If required, engineering models for technologically new or critical equipment may be built and may also be integrated into the Real-Time Testbed to validate functional and electrical interfaces. In such cases, the simulator will be launched in another configuration with the corresponding models disabled.

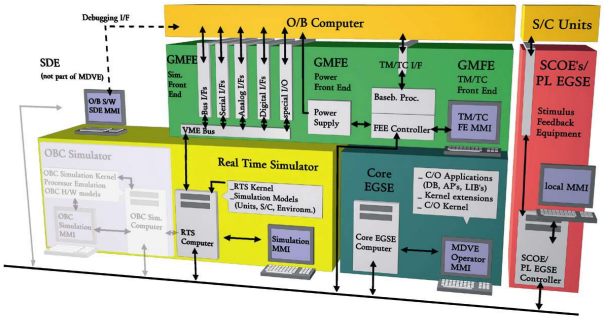


Figure 13. Real-Time Testbed (RTB) [7]

Hence, the transition from the Real-Time Testbed to the satellite level integration and test may be accomplished via several intermediate configurations leading finally to a so-called "FlatSat" testbench, which integrates further hardware such as sensors, actuators and payloads into the loop. All as real hardware integrated units are de-activated in the simulation and being served with stimulation data and commanded via "Special Check-Out Equipment's" (SCOE's). The connection is achieved via so-called "Generic Modular Front-end Equipment" consisting of:

- a TM/TC commanding front-end connecting the OBC to the control console
- a power front-end supplying the OBC with power and
- a simulation front-end routing all OBC I/O to the RTS simulation (S/C environment conditions, orbit dynamics etc.) and the spacecraft equipments.

A major benefit of model-based system development utilizing MDVE is the possibility for early simulated satellite mission operations, using real on-board software in the virtual satellite. By the use of the model based concept each flight hardware unit can be realized by an equivalent software model prior to the availability of the real hardware. This represents an outstanding support to system design qualification and performance verification.

6.2. Application for development of ACS algorithms

Utilizing a model-based system-simulation environment for micro-satellites a new and improved ACS development and verification strategy can be established. In a very convenient and easy accessible environment Matlab Simulink provides comprehensive toolsets for basic development of ACS control loops, but a detailed consideration of the spacecraft specific hardware performance is unlike harder to implement.

In order to optimize the ACS control loops in a second development stage considering the hardware performance of the *Flying Laptop* attitude control sensor's and actuator's, a real-time testbench needs to be setup. Each spacecraft component will successively be integrated into the simulation by a software model of adequate detail in order to provide the particular testbench functionality. The simulator is connected to a Central Control System (CCS), which may be represented by the ESA developed Spacecraft Control and Operation System (SCOS) 2000 for example. The CCS operates the virtual satellite as well as the simulation environment via TM/TC packages, which in most cases are stacked in automated procedures.

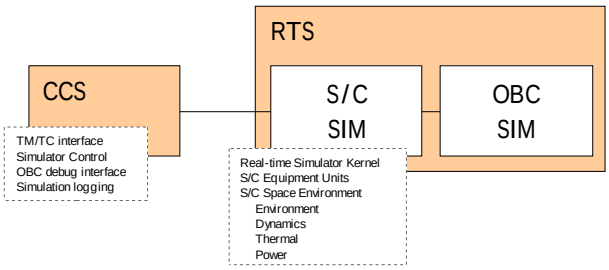


Figure 14. Block diagram of the Software Verification Facility configuration

During design and development a simulation setup with functional represented OBC and later on a full OBC-Simulator is sufficient for attitude control loop development. The configuration is shown in Figure 14. This test setup enables us for a more detailed development of the control algorithms, by means of considering the specific hardware characteristics and performance, well before real flight hardware availability. Moreover for control loop algorithm improvement this closed-loop simulation environment leads to better results than a pure hardware testbench ever can.

Later on in the test and verification phase this configuration will be extended into a hardware-in-the-loop Real-Time Testbed to verify software ↔ software and software ↔ hardware compatibility. In order to close the simulation loop the Real-Time Simulator will remain in the setup stimulating or bypassing the hardware sensors like GPS, Startracker, Gyroscope, Magnetometer or Sunsensors with simulated data (refer to Figure 15)

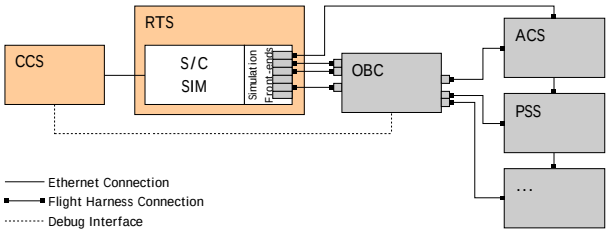


Figure 15. Block diagram of the FlatSat test bench with flight hardware (grey shaded) in the simulation loop

The described model-based development and verification process is a challenging but promising approach, especially for low budget micro-satellite projects. While setting up and strictly qualify the simulation environment in parallel to the Flying Laptop progress and applying it throughout the design and development process, it is important to point out, that the same toolset is also capable for the final requirements verification of the spacecraft. This leads us to a mayor advantage: cost effectiveness and according to this an enabling technology for realizing the demanding objectives of this university project.

7. CONCLUSION

For the calibration concept using LEDs it can be stated, that a concept is designed, which perfectly matches the different requirements of a micro satellite. A very simple concept was presented, which so far seems to provide all needed functionality. The model-based approach for the attitude control algorithms will consider specific flight hardware characteristics. Further the same simulation environment, which is applied for the algorithm development, is also capable for the verification of the whole onboard computer software.

8. REFERENCES

[1] Grillmayer, G. et al.:*Flying Laptop – Micro-Satellite of the University of Stuttgart for Earth Observation and Technology Demonstration*, IAC-04-IAA.4.11.P.08, 55th IAC, Vancouver, Canada, 2004.

[2] Walz, S. et al.: *Payload and Scientific Investigation Objectives of the Flying Laptop*, Selected Proceedings of the 5th International Symposium of the IAA, Berlin, Germany, April 2005, p. 419 – 427, 2005.

[3] Nieke, J. et al.: *Spaceborne Spectrometer Calibration with LEDs*, SPIE Vol.4135, 2000.

[4] Schubert, F.: *Light-Emitting-Diodes*, Cambridge University Press, Cambridge, ISBN0521823301, 2003.

[5] Grillmayer, G., Hirth, M., Huber, F., Wolter, V.: *Development of an FPGA Based Attitude Control System for a Micro-Satellite*, AIAA-2006-6522,

AIAA/AAS Astrodynamics Specialist Conference, Keystone, CO, USA, 21-24 Aug. 2006.

[6] Eickhoff, J.; Falke, A.: Röser, H.-P.: *Model-based Design and Verification – State of the Art from Galileo Constellation down to small University Satellites*. Spain, Valencia: 57th International Astronautical Congress (IAC), October 2-6, 2006.

[7] Eickhoff, J.: *Systemsimulation in der Satellitenentwicklung I&II ─ (System simulation in Satellite Engineering I&II)*. Germany, Stuttgart: Annually lectures at Institute of Space Systems, Universität Stuttgart.

Available online at www.sciencedirect.com
ScienceDirect
PERGAMON
Acta Astronautica 61 (2007) 383–390
ACTA ASTRONAUTICA
www.elsevier.com/locate/actaastro

Model-based design and verification—State of the art from Galileo constellation down to small university satellites

Jens Eickhoff[a,*], Albert Falke[b], Hans-Peter Röser[b]

[a] *EADS Astrium GmbH, Immenstaad, Germany*
[b] *Universität Stuttgart, Stuttgart, Germany*

Available online 29 March 2007

Abstract

Since 2001 EADS Astrium GmbH has implemented a model-based system simulation infrastructure for support of spacecraft development, onboard software verification and spacecraft design validation. Corresponding thereto an engineering process has been established, allowing development and full design validation of spacecrafts without necessity of engineering models on spacecraft level. This simulation infrastructure is called "model-based development and verification environment" (MDVE). Major benefit of model-based system development applying MDVE is the early possibility for simulated satellite mission operations, using real onboard software in the virtual satellite, even before availability of real hardware. This represents an outstanding support to system design qualification and performance verification. MDVE technology application, however, is not limited to commercial and large-scale spacecrafts. In a close partnership with Universität Stuttgart, EADS Astrium also sponsored an MDVE installation to the Institute of Space Systems (IRS) for the small-satellite program *Flying Laptop*, which is the first micro-satellite under development of IRS. This paper presents the span of benefits of model-based engineering with MDVE in application from small satellites up to spacecraft constellations like Galileo. The MDVE hardware and software concepts, typical testbench constellations and satellite design process characteristics are presented with application examples given both from the micro-satellite *Flying Laptop* as well as from the Galileo program.

1. Introduction

1.1. Traditional approach for satellite design and development

The traditional approach to the functional, respectively, electrical development and verification of satellites usually included a dedicated electrical system model in addition to the flight model. This electrical system model also was called "engineering model". The purpose of this model was to develop and validate spacecraft test procedures and to perform the functional and electrical qualification or at least a prequalification.

In line with this philosophy, the general tendency in the past was to include as much real hardware in the loop as possible, to attain realistic verification results. Also for supplementary verification setups such as the onboard software verification facilities (SVFs), hardware was included at least for the onboard processor to run the flight software.

1.2. Model-based development approach

Two major tendencies since quite some time were driving to change the paradigm for satellite design and

* Corresponding author.
E-mail addresses: jens.eickhoff@astrium.eads.net (J. Eickhoff), falke@irs.uni-stuttgart.de (A. Falke), roeser@irs.uni-stuttgart.de (H.-P. Röser).

0094-5765/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.actaastro.2007.01.027

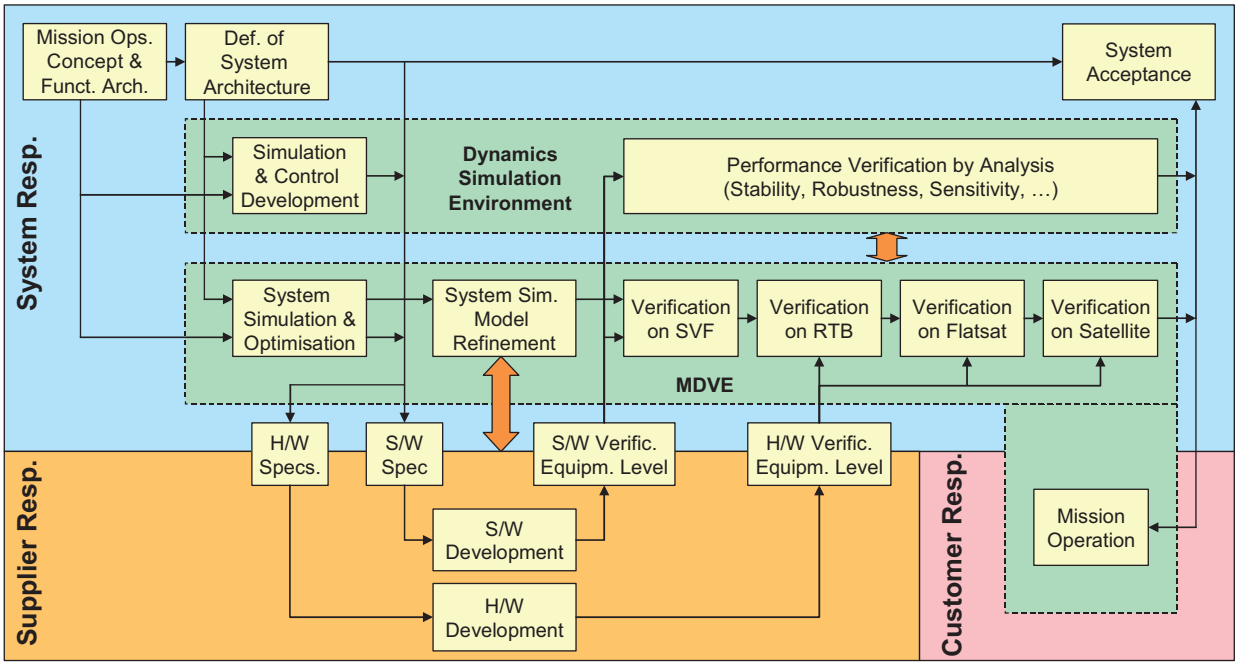


Fig. 1. Flow of major activities for model-based development.

development. First, it is necessary to reduce cost due to shrinking budgets of institutional and commercial customers. Second, the advancements in the information technology—both hardware and software—give new perspectives to the design and development process (Fig. 1).

Since 2001 EADS Astrium GmbH has implemented a model-based system simulation infrastructure to support spacecraft development, onboard software verification and spacecraft design validation without necessity of engineering models on spacecraft level. Corresponding thereto an engineering process has been established. This simulation infrastructure is called "model-based development and verification environment" (MDVE) [2].

Major benefit of model-based system development applying MDVE is the early possibility for simulated satellite mission operations, using real onboard software in the virtual satellite, even before availability of real hardware [3–5].

The model-based MDVE approach represents an outstanding support to system design qualification and performance verification. The infrastructure provides a tool infrastructure allowing setup of S/C models ranging from early pure virtual simulations via hybrid testbenches up to full FlatSat configurations. This largely minimizes cost for hardware models; however, in parallel provides risk mitigation through stepwise verification of onboard SW, onboard HW, flight procedures, etc., first on simulation testbenches and later on HW in the loop configurations. Since over several projects the technology has been proven meanwhile there is a strong tendency, to use purely numerical simulation as this presents advantages such as

- Development of numerical models is quicker and cheaper than hardware mock-ups.
- Numerical models are more easily adaptable to the design modifications.
- Numerical testbenches are easily multipliable in case of satellite project schedule constraints.
- No problems with transport, customs and related logistics.

2. MDVE Technology

2.1. The engineering process

The satellite functional behavior and interaction between onboard equipment is modeled on the basis of the system and equipment specifications. As an output from this block, updated and validated specifications for flight equipment and software can be given to the subcontractors.

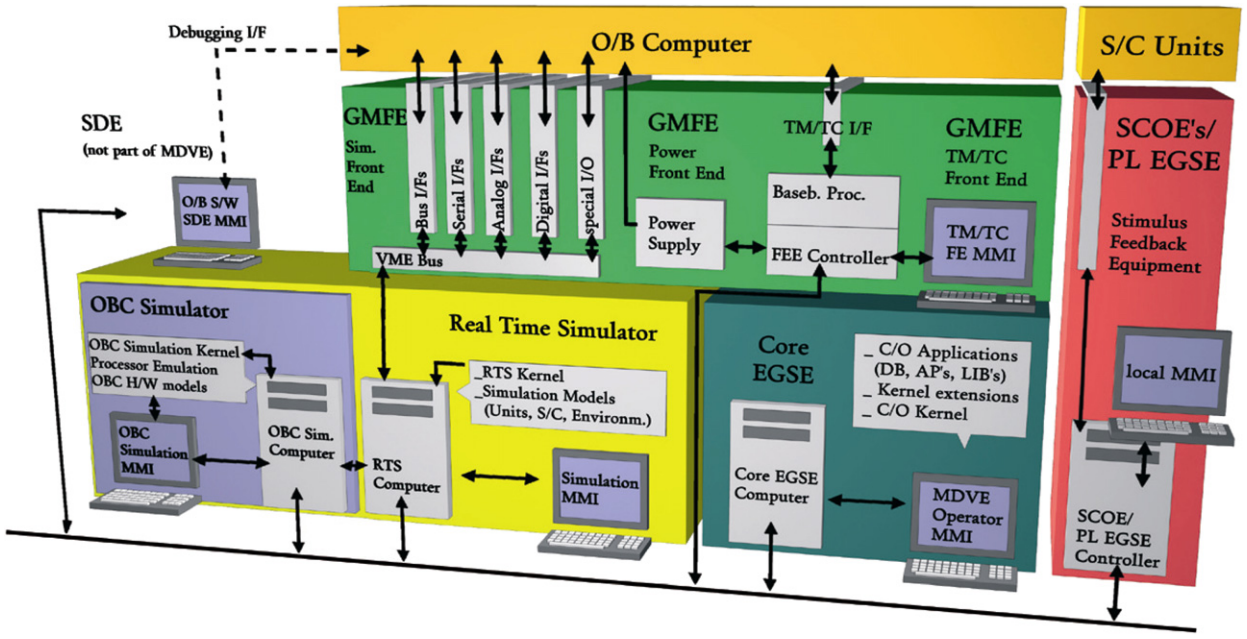


Fig. 2. Building blocks and interfaces of the MDVE construction kit.

With the availability of the onboard computer (OBC) simulation and the refined modeling of the satellite equipment and electrical/functional architecture, the facility for software verification is setup. It supports the low-level software verification at the flight software supplier and the high-level software validation at the prime contractor. From this stage onwards, the SVFs usually are used throughout the satellite life cycle to support debugging and implementation of new releases to the flight software as well as for the development of onboard control procedures.

The next step is to setup the so-called real-time testbed (RTB), which includes a bread board or engineering model of the OBC to enable the hardware/software integration and compatibility verification of the flight software under real hardware conditions. For the functional and behavioral representation of the rest of the satellite, an SVF simulator copy is used.

As outlined above, for technologically new or critical equipment, engineering models may be built. These can be integrated into the RTB to validate functional and electrical interfaces. In such case, the simulator is reconfigured with the corresponding models disabled. Hence, the transition from the RTB to the satellite level integration and test may be accomplished via several intermediate configurations leading finally to a so-called "FlatSat" configuration.

2.2. The MDVE toolkit

The core element of the MDVE is an onboard computer simulator (OBC simulator) and a real-time simulator (RTS) modeling the rest of the S/C plus space environment conditions. In pure simulation setups these simulators (synchronized to each other) are commanded via a control console—in most cases a core EGSE (Fig. 2).

In hardware in the loop testbench setups usually first the OBC is included as hardware, replacing the OBC simulator and being connected to the RTS—now really running under hard real-time conditions to serve the OBC. The connection is achieved via a so-called "generic modular front-end equipment" consisting of:

- a TC/TM commanding front-end connecting the OBC to the control console,
- a power front-end supplying the OBC with power and
- a simulation front-end routing all OBC I/O to the RTS simulation, the spacecraft equipments, S/C environment conditions, orbit, etc.

In the cited FlatSat setups further hardware can be integrated into the loop such as sensors, actuators, payloads, being then deactivated in the simulation and being served with stimulation data and commanding via so-called "special checkout equipment" or SCOEs.

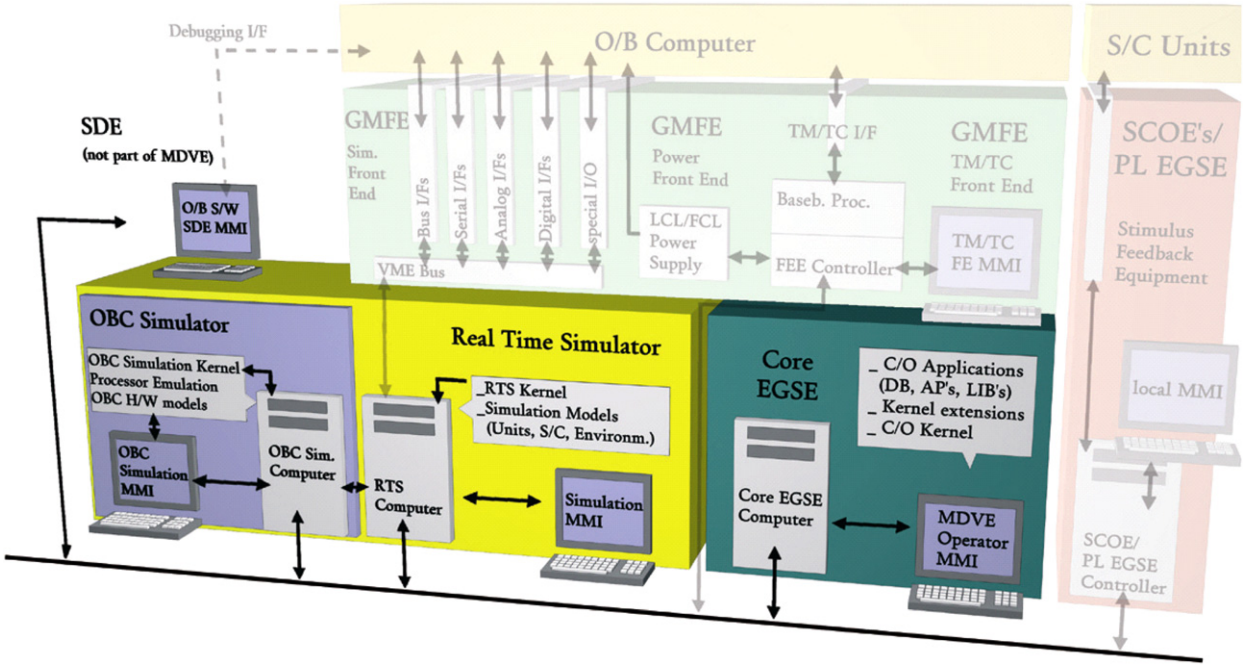


Fig. 3. Software verification facility (SVF).

The next subsections explain the setups in which the different elements are used phase by phase.

2.3. MDVE configurations

As stated above, the MDVE will be used in various project phases for a number of design development and verification tasks in different working environments. The most important MDVE standard configurations are:

- development SVF (DEV-SVF),
- software verification facility (SVF),
- real-time testbed (RTB),
- extended real-time testbed (FlatSat),
- spacecraft simulator for the mission control center.

The following sections give a short overview on each of the MDVE configurations.

2.3.1. Software verification facility

The SVF setup is illustrated in Fig. 3. The facility consists of a dedicated simulator for the OBC. The OBC is modeled functionally to a level of detail that real BIOS, RTOS and onboard application SW (OBSW) can be loaded and executed on it. The RTS is connected as slave to the OBC simulator. Both simulators and the

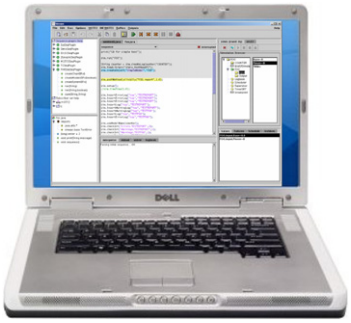


Fig. 4. SVF for onboard SW development (DEV-SVF configurartion).

"virtual simulated S/C" are commanded via the control console (Fig. 4).

SW design of the SVFs is based on UML/C++, both for the simulator infrastructure as well as for the modeling of the S/C itself.

Typically 2 setups of such SVFs exist nowadays:

- One is commanded by a more lightweight Java-based control console for the OBSW developers. It is not optimized for execution of full flight procedures requiring excessive numerical performance. Therefore it is executable on simple PCs and laptops.
- The second, full SVF, is optimized for execution of full flight procedures and typically is controlled by

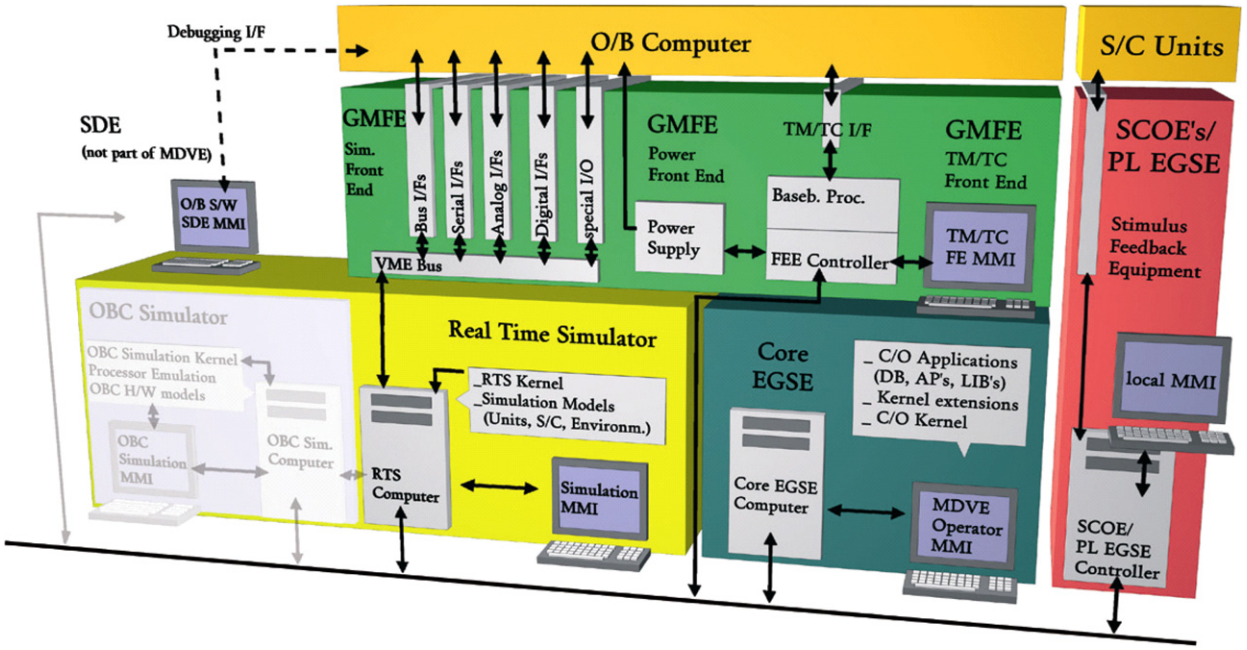


Fig. 5. Real-time testbed (RTB).

a full database-driven core EGSE and is running on high-performance Linux workstations. The purpose of the SVF is to test the OBSW. Hard real-time conditions are not required since no hardware is in the loop.

2.3.2. *Real-time testbed*

The configuration of the RTB is shown in Fig. 5. The main difference of this setup compared to those shown above is the inclusion of a real hardware model of the OBC. It replaces the OBC simulator by real OBC hardware in the loop. The core EGSE remains the overall RTB operator interface. In this configuration the RTS has to serve the I/O from the OBC in hard real time over all bus channels and therefore it is running on a real-time operating system platform (VxWorks) using a VME-bus-based simulator front-end for data I/O between simulator and OBC.

The purpose of the RTB is to test the OBSW together with the simulated satellite, having the software loaded into the real OBC—e.g. for hardware timing tests.

2.3.3. *FlatSat*

Starting from the RTB configuration, the MDVE can be configured to form the satellite FlatSat, which is illustrated in Fig. 9. Whenever a real hardware unit will come into the loop it will be accompanied by its SCOE, and its models will have to be removed from the RTS and the GMFE. The need of an RTS on this integration level depends on the project-specific test philosophy.

The finally tested flight hardware and software then can be mounted from the FlatSat configuration into the spacecraft structure (see Fig. 10).

2.3.4. *Simulator for the mission control center*

The final application of the simulator within the overall satellite life cycle is to support developing flight operation procedures, mission operators training and to support mission operation especially in pretesting OBSW patches.

The OBC simulator and RTS from SVF configuration are connected to the satellite ground control center instead of the control consoles used in DEV-SVF or SVF configurations. This is possible since the communication protocols are in both cases based on the same ESA standard.

3. Applications in industry

The model-based development technology for the first time was applied in ESA's CryoSat program. In the meantime it is established at EADS Astrium as standard technology, being applied in the projects GOCE (ESA), TerraSAR (DLR), Aeolus (ESA) and the LISA

Fig. 6. Full SVF (example Aeolus).

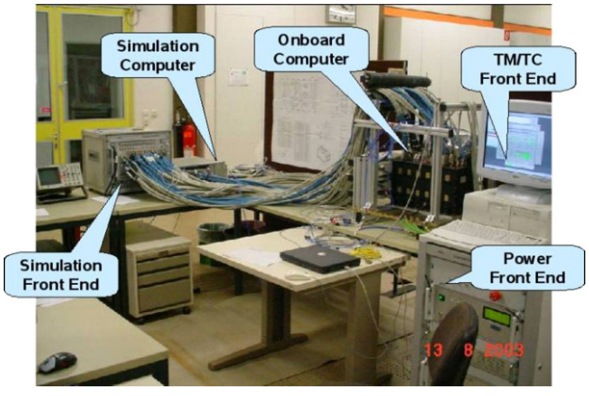


Fig. 7. Real-time testbed (example CryoSat).

pathfinder mission (ESA) with extreme requirements towards software performance and numeric precision (Figs. 6–11).

The latest derivates of the technology are applications in Europe's navigation strategy program Galileo, where both the space segment level as well as the avionics subsystem of the Galileo in-orbit-verification satellites are developed applying MDVE technology.

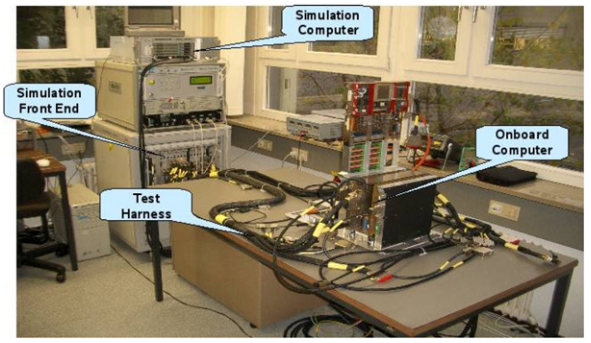


Fig. 8. Real-time testbed (example TerraSAR).

Fig. 9. CryoSat FlatSat configuration.

4. Applications in science

4.1. Industry–science partnership

MDVE technology application, however, is not limited to commercial and large-scale spacecrafts. In a close partnership with Universität Stuttgart, EADS Astrium also sponsored an MDVE installation to the Institute of Space Systems (IRS) for the small-satellite program *Flying Laptop* (Fig. 12) [1], which is the first micro-satellite under development of IRS.

While the IRS is supported by EADS Astrium with the MDVE satellite development environment as well as via a 2 semester lecture on technology and internal backgrounds of this simulation-based approach [6], industry's objective in this partnership is to have access to students and PhD students being trained to state-of-the-art engineering technology and processes.

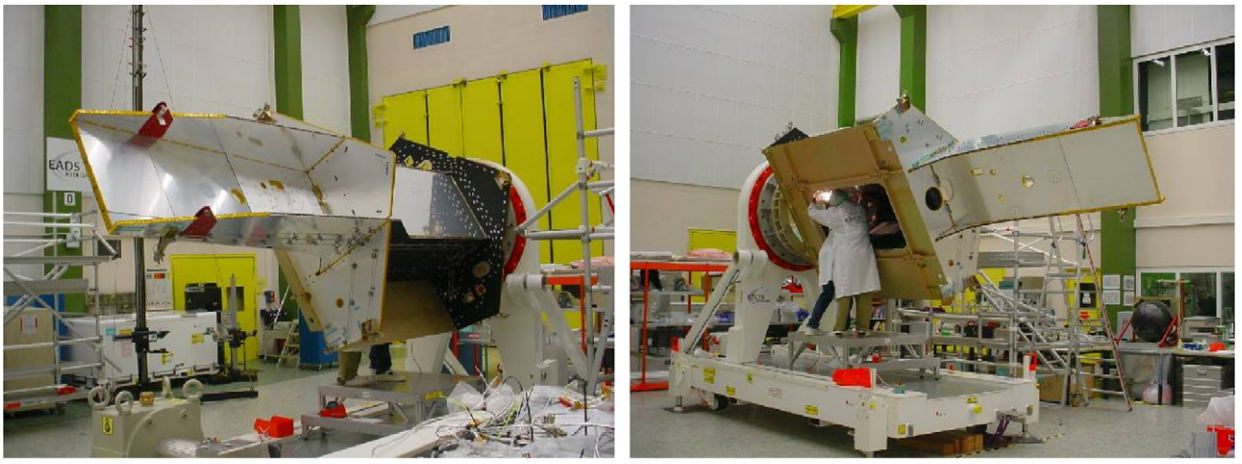


Fig. 10. Mounting tested flight equipment from FlatSat into structure.

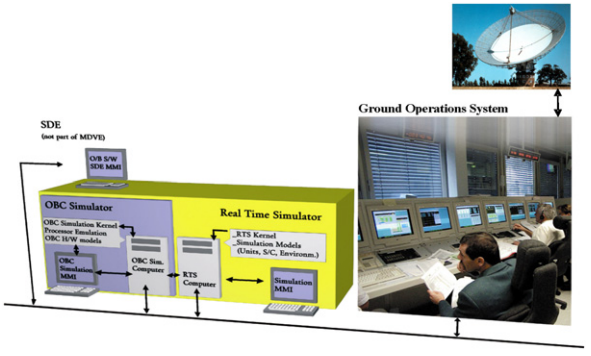


Fig. 11. Satellite simulator for the mission control center.

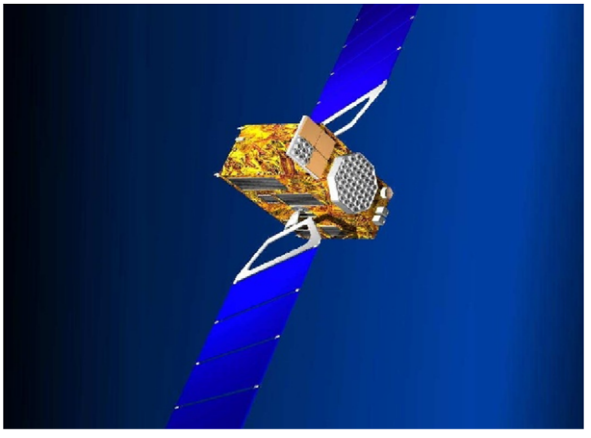


Fig. 12. Galileo IOV satellite.

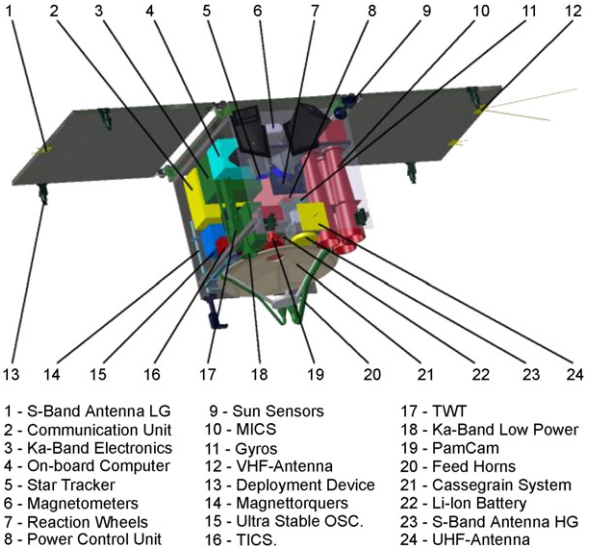


Fig. 13. Design of the *Flying Laptop*.

4.2. IRS satellite and associated technology development

The *Flying Laptop* will be the first micro-satellite of the Universität Stuttgart small-satellite program. The primary mission objective is to demonstrate and qualify new small-satellite technologies for the future projects. As a secondary objective, multiple scientific earth observation experiments are planned.

The satellite body has a cubical shape with an edge length of 60 cm and a mass of less than 100 kg. Fig. 13 shows the general design of the satellite including its components. The launch as a piggyback payload is planned for the end of 2007 or in 2008. A polar, sun-synchronous, low earth orbit below 1000 km is being pursued. As scientific payload the satellite is equipped with a 3-camera system (VIS/NIR), a thermal infrared (TIR) camera and a Ka-band communication system. The last two are intended to make dual use of a cassegrain mirror system.

Several research technologies will be implemented aboard, such as a highly agile attitude control system operating in target pointing mode for image acquisition. Furthermore the OBC is based on a reconfigurable, redundant and self-controlling field programmable gate array (FPGA) instead of classic micro-processor technology.

Both topics mentioned above require an extensive on-ground-verification test campaign before launch to assure the intended functionality. The application of a comprehensive verification environment like the MDVE with its ability for reliable system-wide tests is new to micro-satellite projects, but it is one of the enabling technologies for proving the required attitude control system accuracy.

In this context a software-based functional verification system reduces the checkout environment complexity and huge costs can be saved.

5. Conclusion

As a conclusion it can be summed up that the model-based development approach for satellites

- with high-performative development tools like UML-based simulator design,
- highly sophisticated simulators—including OBC models for onboard software testing and
- the possibility for stepwise exchange of simulated models by hardware in the loop in these testbenches up to a FlatSat configuration

leads to a cost efficient and schedule optimized development approach necessary for both successful industry programs as well as successful technology research missions.

References

[1] G. Grillmayer, A. Falke, H.-P. Roeser, Technology demonstration with the micro-satellite Flying Laptop (IAA-B5-1402), in: The fifth IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, April 4–8, 2005.

[2] J. Eickhoff, J. Flemmig, Model-based development and verification environment, in: Data Systems in Aerospace (DASIA)'2002 Conference, Dublin, Ireland, May 13–17, 2002.

[3] J. Eickhoff, R. Hendricks, J. Flemmig, Model-based development and verification environment, in: The 54th International Astronautical Congress (IAC), Bermen, Germany, September 29–October 1, 2003.

[4] J. Eickhoff, Model-based development and verification environment, in: The Sixth NASA/ESA Workshop on Project Data Exchange, Hosted by EADS Astrium GmbH, Friedrichshafen, Germany, April 20–23, 2004.

[5] R. Hendricks, J. Eickhoff, The significant role of simulation in satellite development and verification—Die Schlüssenrolle von Simulationstechnik in Satellitenentwicklung und Test, Aerospace Science and Technology 9 (3) (2005) 273–283.

[6] J. Eickhoff, Systemsimulation in der Satellitenentwicklung I&II—(System simulation in satellite engineering I+II). Annual lectures at Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany.

PRELIMINARY SYSTEM SIMULATION ENVIRONMENT OF THE UNIVERSITY MICRO-SATELLITE *FLYING LAPTOP*

Alexander Brandt[(1)], Ivan Kossev[(1)], Albert Falke[(1)], Jens Eickhoff[(2)], Hans-Peter Roeser[(1)]
[(1)] Universität Stuttgart, Pfaffenwaldring 31, 70569 Stuttgart, Germany,
Phone: +49 711 685 69637, Email: falke@irs.uni-stuttgart.de
[(2)] EADS Astrium GmbH, 88039 Friedrichshafen, Germany,

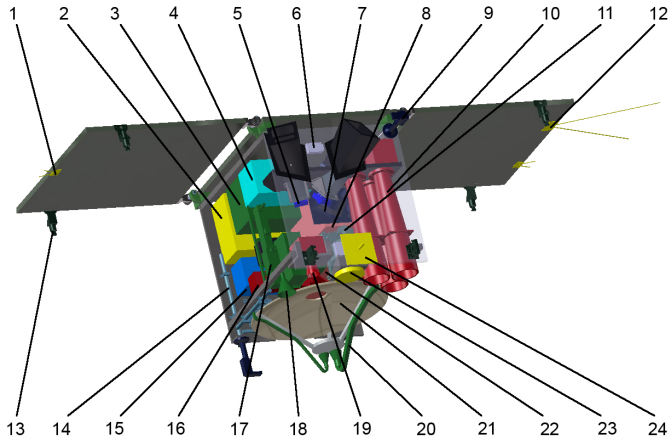
ABSTRACT

The Institute of Space Systems at the Universität Stuttgart builds a micro-satellite for the purpose of technology evaluation and scientific experiments. The complexity of the project and its limited human resources require efficient modern development and engineering techniques. To support the satellite development process, EADS Astrium Friedrichshafen has sponsored a Model-based Development and Verification Environment infrastructure, to be used for test and design verification. This paper focuses on the set-up of the Model-based Development and Verification Environment simulator, in particular on the configuration of the simulated satellite environment, the creation of equipment models and operational facilities to control the simulated satellite and the simulator. Orbit and thermal simulations were conducted to demonstrate the functionality of the environment setup under consideration.

1. INTRODUCTION

Since 2003 the Institute of Space Systems of the Universität Stuttgart carries out the Stuttgart Small Satellite Program whose first spacecraft is the *Flying Laptop* (Figure 1). The *Flying Laptop* project's objectives are technology demonstration and Earth observation [1, 2]. The complexity of the project requires efficient modern development and verification techniques. To support the satellite development process, EADS Astrium Friedrichshafen has sponsored a Real-time Simulator (RTS) infrastructure, which is used for test and design verification.

The Real-time Simulator is part of Astrium's Model-based Development and Verification Environment (MDVE) [3, 4], which is a real-time capable functional simulation environment. The



1 S-band Antenna LG
2 Communication Unit
3 Ka-band Electronics
4 On-board Computer
5 Star Tracker
6 Magnetometers
7 Reaction Wheels
8 Power Control and Distribution Unit
9 Sun Sensors
10 Multispectral Image Camera System (MICS)
11 Fiber Optical Gyroscopes
12 VHF Antenna
13 Deployment Device
14 Magnetic Torquers
15 Ultra Stable Oscillator
16 Thermal Image Camera System (TICS)
17 Traveling Wave Tube
18 Ka-band Low Power
19 Panoramic Camera
20 Feed Horns
21 Cassegrain System
22 Li-Ion Battery
23 S-band Antenna HG
24 UHF Antenna

Figure 1: Design of the *Flying Laptop*.

MDVE configuration used at the Institute of Space Systems includes a Real-time Simulator and a Core Electrical Ground Support Equipment (EGSE). The latter is required to command the RTS and the simulated satellite. In the current MDVE implementation the Core EGSE is substituted by the Mission Control System (MCS). The Real-time Simulator consists of four parts (Figure 2):

- the On-board Computer (OBC) Simulator,
- the satellite equipment models,
- physical models, which simulate the spacecraft dynamics or thermal aspects,
- and the simulator kernel, which includes important low-level simulator functions like the scheduler, the numerical solver and the Telemetry (TM)/Telecommand (TC) handler.

Prior to this work, the Real-time Simulator at the Institute of Space Systems consisted of an

- initial version of the OBC and Power Subsystem models,
- as well as of most of the Attitude Control System (ACS) sensors.

This paper discusses the extension of the existing MDVE RTS to a first functional state. This includes the following steps:

- As a basis for the integration of ACS control loops, ACS actuator models must be created and the Environment and Dynamics Physical Models must be verified.
- For conducting thermal simulations, a S/C thermal model must be created and verified. Additionally, thermal control system hardware models must be implemented.
- To enable simulation data processing and evaluation, the RTS as well as the simulated S/C must be connected to a MCS.

In this paper the implementation of these steps is presented.

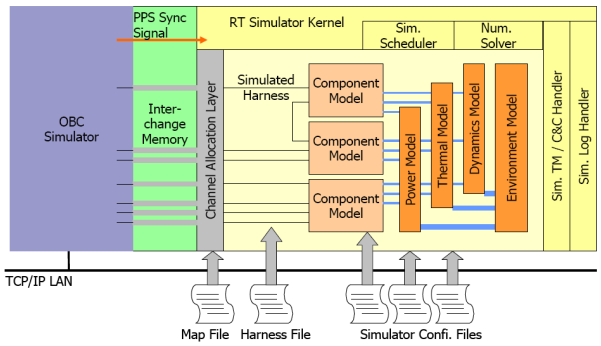


Figure 2: The Real-time Simulator structure.

2. IMPLEMENTATION

2.1 Integration of the Environment and Dynamics Propagator into the RTS

The environment and dynamics models have been sponsored by EADS Astrium Friedrichshafen and implement the RTS Environment and Dynamics Propagator. Quantities, which are calculated by the propagator, are accessible for all RTS models via an environment and dynamics interface. For example a sensor equipment model is interconnected to the propagator via this interface to acquire data which is then transmitted to the OBSW as measured sensor data. The ACS actuator models report their induced forces, torques and angular momentum to the propagator as input for the integration of the S/C equations of motion. Figure 3 shows which propagator quantities are exchanged with the magnetic torquer and reaction wheel models. The Environment and Dynamics Propagator implements the environment models given in Table 1. These models may be enabled or disabled (except for the sun and moon gravitational effects) providing comparable functionality as the High Precision Orbit Propagator (HPOP) of the Satellite Tool Kit (STK) [5].

At simulator initialization, the propagator is characterized by data from the RTS configuration

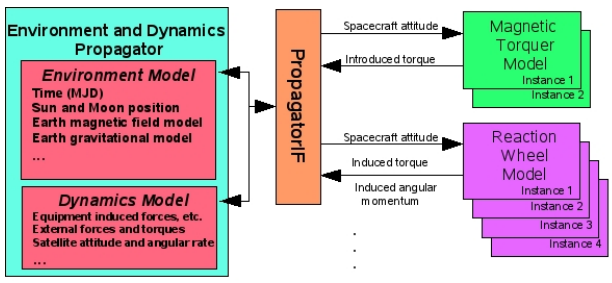


Figure 3: Environment and dynamics infrastructure.

files as follows:

- The environment models from Table 1 are initialized.
- The simulation start time is set.
- Initial position, velocity and attitude are set.
- Satellite parameters like mass, moment of inertia matrix and geometry parameters are set.

The position and velocity accuracy of the Environment and Dynamics Propagator has been verified by comparing simulated orbits with data acquired by STK [6]. The simulations show that the propagator's accuracy agrees well with the accuracy of STK's HPOP. Figure 4 illustrates the time dependency of the Longitude of Ascending Node acquired by an STK and an MDVE simulation. Data has been logged in 10s intervals, which affects the calculation precision.

Figures 5 and 6 represent a comparison of the orbit inclination and altitude acquired by both simulation environments. These show good agreement, which certifies the Environment and Dynamics Propagator as suitable for ACS and orbit propagation within this project.

2.2 Integration of the Thermal Propagator into the RTS

The Thermal Propagator consists of a thermal integrator, sponsored by EADS Astrium, and a nodal thermal model. The propagator interacts

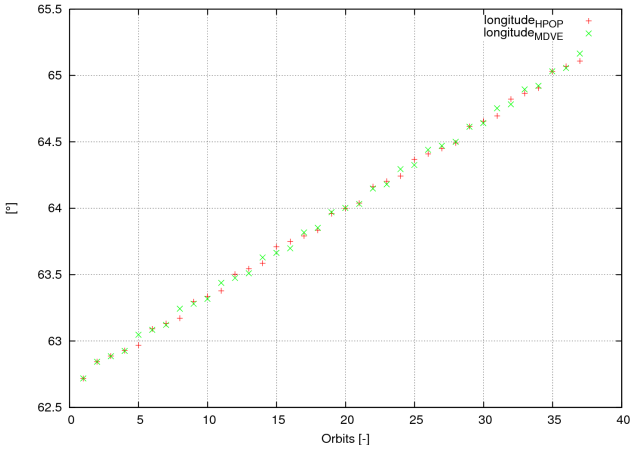


Figure 4: Time dependency of the orbits longitude of the ascending node for the Flying Laptop nominal orbit. Comparison between the HPOP and the MDVE simulations.

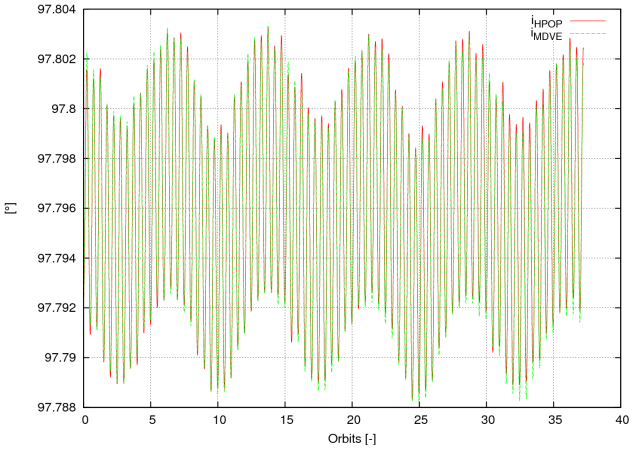


Figure 5: Orbit inclination time dependency for the *Flying Laptop* nominal orbit. Comparison between the HPOP and the MDVE simulations (i_{NOM}= 97.79°).

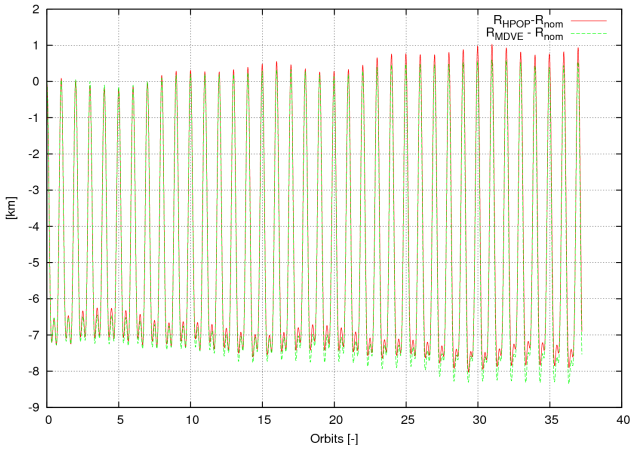


Figure 6: Time dependency of the orbit altitude deviation for the *Flying* Laptop nominal orbit. Comparison between the HPOP and the MDVE simulations.

Table 1: Environment and Dynamics Propagator environment models.

Model type	Model name
Earth atmospheric model	fixed density or interpolation table data
Solar and albedo flux models	interpolation table data
Earth magnetic field model	IGRF10
Earth gravitational field model	JGM3
Sun and moon gravitational effect	no model name

with the equipment models via a special Thermal Interface and integrates a system of energy conservation equations whose inputs are the solar, Earth infra-red and albedo heat fluxes as well as the dissipated power provided by the equipment models. The current implementation of the Thermal Propagator supports only orbit-averaged solar, Earth infra-red and albedo heat fluxes, which are invariant with respect to the S/C attitude. The propagator's outputs are the nodal temperatures.

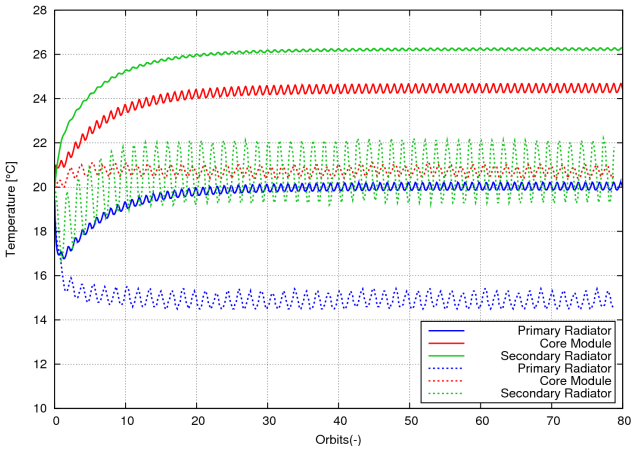


Figure 8: Propagation of a temperature throughout the RTS to the MCS.

Thermal Model

The thermal model describes the thermal properties of the S/C and is therefore the coupling between the general purpose Thermal Propagator and the S/C under consideration. The thermal model is of a nodal type and consists of a reduced number of nodes to ensure low numerical complexity and thus the real-time capability of the RTS. In case of the *Flying Laptop*, 33 of these nodes model the S/C's equipment and structure and one node represents the space environment [7]. The reduced number of nodes is achieved by assigning more than one piece of equipment to one thermal node. To reduce modeling errors, attention has been paid that this node assignment is done for adjacent equipment. The node coupling network of the *Flying Laptop* MDVE thermal model is depicted in Figure 7 for a constant sun-oriented S/C attitude. The nodal discretization of the S/C causes that the resulting thermal conductive couplings are lower than the real ones [8]. Since the propagator is invariant with respect to the S/C's attitude, the nodal network cannot be adjusted to attitude changes during simulation. The solid lines represent conduction couplings and the dashed ones represent radiation couplings. The couplings' values are calculated using analytical approximation expressions [8]. Only radiation conductors, which model incoming or outgoing heat fluxes, are shown. The internal radiation couplings are omitted to keep the illustration concise. The incoming solar (S), earth infra-red (E) and albedo (A) heat fluxes are represented by three virtual nodes, which are not part of the nodal thermal model but are displayed for reasons of clarity. In contrast, node 34 absorbs the outgoing heat fluxes and is part of the nodal model. Thermal simulations are conducted and verified by comparison with I-deas NX software simulations [9]. Figure 8 shows the temperature time dependency of the Payload, Core, and Service Modules. The following differences can be observed:

- The temperatures in the MDVE simulation are higher than in the I-deas NX simulation, because of the MDVE thermal model's lower resulting thermal conductive couplings.
- Generally, the MDVE features lower temperature amplitudes due to the orbit-averaged thermal loads in this simulation.
- The temperature amplitude of the Secondary Radiator in the I-deas NX simulation is much larger than the one in the MDVE simulation. It is assumed that this is also caused by the orbit-averaged thermal loads.

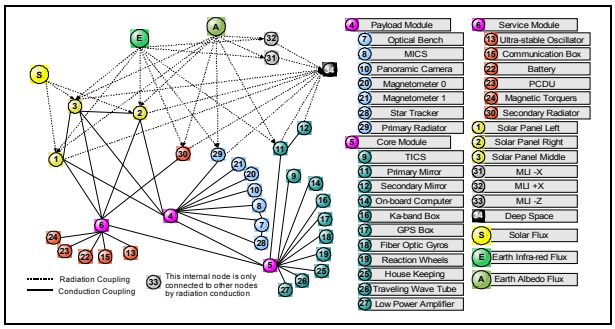


Figure 7: The *Flying Laptop* nodal thermal model.

- A gap between the temperatures of the MDVE Secondary Radiator and the temperatures of the Core Module can be observed, which is absent in the I-deas NX simulation. It is assumed that this is caused by the lower resulting thermal conductive couplings.

Since the main objective of MDVE is system simulation and not detailed thermal analysis, the results of the conducted simulations confirm the validity of the created thermal model despite the temperature differences.

2.3 Equipment Models

The S/C is represented in the RTS by a number of equipment models, each of them modeling specific equipment hardware. Each piece of hardware is implemented by a single C++ class with a number of instances corresponding to the equipment's occurrence in the S/C. Initially, a Unified Modeling Language class diagram of the hardware equipment is created. This diagram contains information about the number of instances to be created from this class, about the number and type of the lines of the simulated hardware harness as well as information about the model's monitoring variables, which will later be accessible for monitoring and manipulation. From this UML diagram an automatic source code generation tool creates the hardware model class framework. This equipment modeling approach results in identically organized and functioning class instances, representing multiple equipments of a certain type. For example, in the *Flying Laptop* there are four identical momentum wheels, mounted in a tetrahedral configuration. From a modeling point of view, the wheels' functionality is identical and the only difference between them is their orientation and state at a certain point of time. By using a base class, the equipment's behavior is implemented only once, while each instance is characterized at simulator initialization via Extensible Markup Language (XML) configuration files. The XML files contain mission scenario data, which is acquired from the project's database. The model's operation consists in the execution of the equipment functional model and interaction with the remaining equipment and

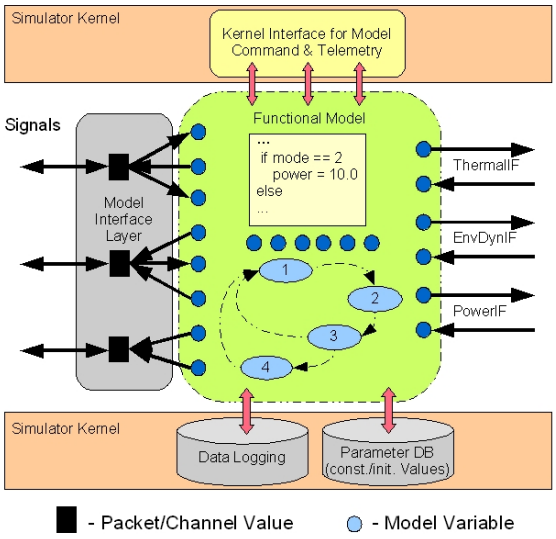


Figure 9: Equipment model diagram.

physical models, as well as with the RTS kernel (Figure 9). The simulator kernel has direct access to the model's system variables (e.g. the model's clock) and is responsible for the logging of model specific data. The equipment's functional model is executed by a fixed-increment time advance mechanism (the simulator scheduler), whereby the model's state is propagated. The functional model consists of the discrete and the continuous models parts, the first one modeling the equipment's mode transitions and the second one calculating continuous quantities. Signals (TM/TC, power supply) are transmitted via the simulated model harness and data is exchanged with the propagators via their corresponding interfaces. As an example of how an equipment model operates, the operation of the reaction wheel model is explained in the following.

Reaction Wheel Operation

The Reaction Wheel model class [6] is instantiated four times to realize the four reaction wheels mounted in the *Flying Laptop*. At simulator initialization, the model is characterized by parameters for:

- the wheel momentum of inertia,
- the nominal supply voltage,
- the maximal motor current, torque and speed,
- the wheel orientation in S/C body frame,
- time constants,
- and the model's initial operational mode.

The Reaction Wheel's functional model is realized by a discrete model part, which models the equipment's hardware and software states as well as by a continuous part, which models the Reaction Wheel's dynamics. Each time the model is invoked, its discrete model part acquires the model's input:

- the supply voltage provided via two power lines by the Power Control and Distribution Unit,
- the equipment's temperature,
- and the induced failure modes.

Having completed this procedure, enough data is available to determine the equipment operational mode. The model's communication reception line is read out and the received commands are processed. Telemetry is sent to the OBC upon request via the model's transmission line and then the model's continuous part is executed. The latter integrates an adapted discretized Reaction Wheel SIMULINK model, developed during Phase A of the project. This model calculates, among others, the wheel momentum, the induced torque and the consumed power, which are finally reported to the Environment and Dynamics Propagator and the Thermal Propagator respectively. This sequence of operations (except for the model's initialization) is cyclically initiated by the scheduler to implement the models state propagation in time.

2.4 Communication

The communication between the system simulator and the MCS consists of two main aspects: a connection between the simulated satellite and the MCS making the communication with the spacecraft possible during simulation, and a second connection between the RTS and the MCS, which is used to command the RTS. Thus the MCS is able to both operate the simulated satellite and to control the RTS (Figure 10). The RTS allows failures to be induced into the simulation to test the On-board Software (OBSW) performance in case of malfunctions [4].

In conventional Software Verification Facility (SVF) implementations a dedicated OBC simulator exists [3], which features an integrated TM/TC front end for communication with the

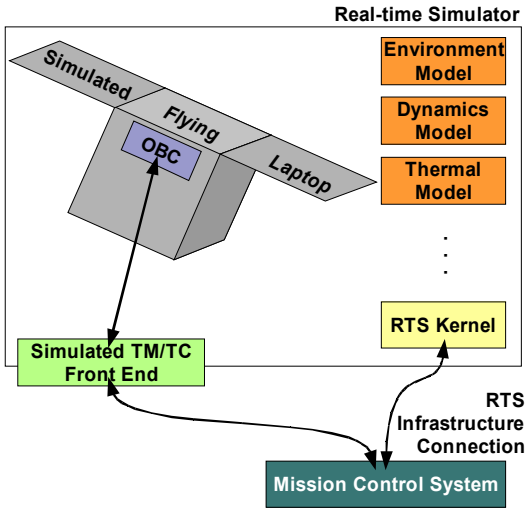


Figure 10: Data links between the MDVE simulation and the MCS.

MCS. The front end directly reads TM from and writes TC into the registers of the emulated OBC [4]. Since the Field Programmable Gate Array (FPGA) OBC of the *Flying Laptop* prevents Direct Memory Access (DMA), a qualitatively different approach for the TM/TC front end integration is developed [7]. The TM/TC will be transferred between the OBC model and a special equipment model, called "TM/TC adapter". Together with a separate TM/TC task, running parallel to the equipment simulation, the simulated TM/TC front end of the *Flying Laptop* RTS is realized. For commanding and telemetry processing, the European Space Agency's SCOS 2000 software is used. SCOS's capability of handling packets according to the Consultative Committee for Space Data Systems (CCSDS) standard requires at present the simulated TM/TC front end to encapsulate the *Flying Laptop* TM/TC packets using the Packet Utilization Standard (PUS). For interfacing the MCS with the RTS and the simulated TM/TC front end, a program called SCOS RTS Proxy is used. It translates pure PUS packets into a format understandable for SCOS-2000. The proxy's functions consist in:

- TM encapsulation
  - TM PUS packets into TM Transfer Frames
  - TM Transfer Frames into Network Controller and Telemetry Router System (NCTRS) TM Data Units

- TC extraction
  - TC Transfer Frames from NCTRS TC Data Units
  - TC PUS packets from TC Transfer Frames
- Routing of TC PUS packets to either the RTS kernel or to the simulated TM/TC front end.

Since the creation of the simulated TM/TC front end, the RTS is capable of handling PUS packets. The creation of the SCOS RTS Proxy enables the communication between the simulator (RTS kernel, simulated TM/TC front end) and SCOS-2000. The following example demonstrates the simulator data flow from the OBC to the MCS.

2.5 Simulator Data Flow

As stated above, the OBC interfaces with the simulated TM/TC front end similarly to an equipment model. The resulting data flow is illustrated by the propagation of temperature data to the MCS (Figure 11). For each scheduled time step, the temperature of a thermal model's node is calculated by the Thermal Propagator and exposed within the Thermal Interface. The House Keeping (HK) unit, an equipment model which interfaces with the temperature sensors, queries the temperature from the Thermal Interface and converts it into a measured voltage, since the real temperature sensors on-board the *Flying Laptop* will also output voltages. Then the HK sends these cyclically to the OBC, which analyzes this data, creates the housekeeping TM packets, and sends them to the TM/TC adapter where they are written into a shared memory area. From the shared memory they are instantly read out by the TM/TC task, which forwards them to the SCOS RTS Proxy. The TM PUS packets are encapsulated into NCTRS TM Data Units by the proxy and sent via the Local Area Network from the Simulation Computer to the SCOS-2000 Packetizer, running on the MCS Computer.

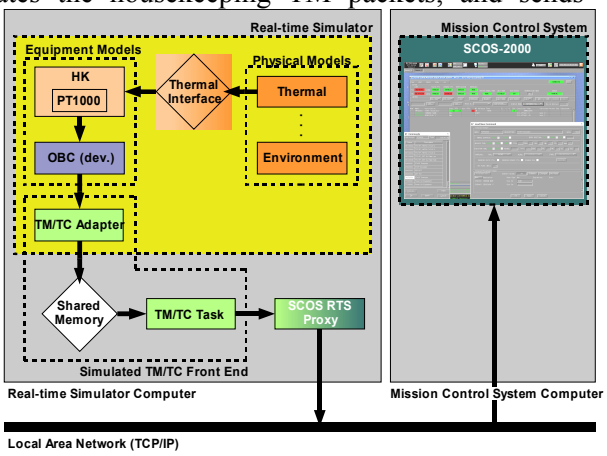


Figure 11: Propagation of a temperature throughout the RTS to the MCS.

3. CONCLUSION

As a conclusion, it can be summed up that

- the creation of actuator models for the ACS, the communication equipment and the thermal control subsystems,
- as well as the integration of the environment, dynamics and thermal physical models,
- and the setup of the MCS and its connection to the simulator

has significantly contributed to the completion of the *Flying Laptop* MDVE first functional state and has paved the way to the Real-time Testbed simulator configuration.

4. REFERENCES

[1] Falke, A.; Grillmayer, G.; Walz, S.; Hesselbach, F.; Eickhoff, J.; Roeser, H.-P.: *LED in-flight calibration and model-based development of ACS algorithms for the university micro-satellite* Flying Laptop. Italy, Chia Laguna: Small Satellite Systems and Services - The 4S Symposium, 25 - 29 September 2006.

[2] Grillmayer, G.; Hirth M.; Huber F.; Wolter, V.: *Development of an FPGA Based Attitude Control System for a Micro-Satellite*. AIAA-2006-6522, USA, Keystone, CO; AIAA/AAS Astrodynamics Specialist Conference. August 21-24, 2006.

[3] Eickhoff, J.; Falke, A.; Roeser, H.-P.: *Model-based Design and Verification – State of the Art From Galileo Constellation Down to Small University Satellites*. Spain, Valencia: 57th International Astronautical Congress (IAC), October 2-6, 2006.

[4] Eickhoff, J.: *Systemsimulation in der Satellitenentwicklung I&II ─ (System simulation in Satellite Engineering I+II)*. Germany, Stuttgart: Annually lectures at Institute of Space Systems, Universität Stuttgart.

[5] AGI: Satellite Tool Kit. *http://www.stk.com/products/desktopApp/stkFamily/*, last visited on 22.04.2007

[6] Kossev, I.: *Development of Functonal Softwae Models of the ACS Actuators and Integration of an Environment*

and Dynamics Propagator into the Simulation Environment for the Micro-satellite Flying Laptop. Institute of Space Systems. Universität Stuttgart. Diplomarbeit. IRS-07-S03. 2007.

[7] Brandt, A.: *Development of Functional Models for Communication and Thermal Hardware and a Thermal Model for the Flying Laptop Micro-satellite Simulation*. Institute of Space Systems. Universität Stuttgart. Diplomarbeit. IRS-07-S-02. Feb 2007

[8] Gilmore, D., Ed.: *Spacecraft Thermal Control Handbook*. Volume I.: Fundamental Technologies. 2nd Ed., AIAA, Reston, Virginia, 2002.

[9] Putze, U.: *Entwicklung des Thermalkonzeptes und Aufbau der analytischen Thermalmodelle für den* Flying Laptop. Institute of Space Systems. Universität Stuttgart. Diplomarbeit. IRS-06-S-04. Jan 2006