Planungssystem zur Steigerung der Autonomie von Kleinstsatelliten

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
an der Fakultät für Mathematik und Informatik
der Bayerischen Julius-Maximilians-Universität Würzburg

Vorgelegt von: Harald Wojtkowiak (M.Eng.)
aus Gerbrunn

1. Gutachter: Prof. Dr.-Ing. Hakan Kayal

2. Gutachter: Prof. Dr.-Ing. Sergio Montenegro

Würzburg 2018

Zusammenfassung

Der Betrieb von Satelliten wird sich in Zukunft gravierend ändern. Die bisher ausgeübte konventionelle Vorgehensweise, bei der die Planung der vom Satelliten auszuführenden Aktivitäten sowie die Kontrolle hierüber ausschließlich vom Boden aus erfolgen, stößt bei heutigen Anwendungen an ihre Grenzen. Im schlimmsten Fall verhindert dieser Umstand sogar die Erschließung bisher ungenutzter Möglichkeiten. Der Gewinn eines Satelliten, sei es in Form wissenschaftlicher Daten oder der Vermarktung satellitengestützter Dienste, wird daher nicht optimal ausgeschöpft.

Die Ursache für dieses Problem lässt sich im Grunde auf eine ausschlaggebende Tatsache zurückführen: Konventionelle Satelliten können ihr Verhalten, d.h. die Folge ihrer Tätigkeiten, nicht eigenständig anpassen. Stattdessen erstellt das Bedienpersonal am Boden - vor allem die Operatoren - mit Hilfe von Planungssoftware feste Ablaufpläne, die dann in Form von Kommandosequenzen von den Bodenstationen aus an die jeweiligen Satelliten hochgeladen werden. Dort werden die Befehle lediglich überprüft, interpretiert und strikt ausgeführt. Die Abarbeitung erfolgt linear. Situationsbedingte Änderungen, wie sie vergleichsweise bei der Codeausführung von Softwareprogrammen durch Kontrollkonstrukte, zum Beispiel Schleifen und Verzweigungen, üblich sind, sind typischerweise nicht vorgesehen. Der Operator ist daher die einzige Instanz, die das Verhalten des Satelliten mittels Kommandierung, per Upload, beeinflussen kann, und auch nur dann, wenn ein direkter Funkkontakt zwischen Satellit und Bodenstation besteht. Die dadurch möglichen Reaktionszeiten des Satelliten liegen bestenfalls bei einigen Sekunden, falls er sich im Wirkungsbereich der Bodenstation befindet. Außerhalb des Kontaktfensters kann sich die Zeitschranke, gegeben durch den Orbit und die aktuelle Position des Satelliten, von einigen Minuten bis hin zu einigen Stunden erstrecken. Die Signallaufzeiten der Funkübertragung verlängern die Reaktionszeiten um weitere Sekunden im erdnahen Bereich. Im interplanetaren Raum erstrecken sich die Zeitspannen aufgrund der immensen Entfernungen sogar auf mehrere Minuten. Dadurch bedingt liegt die derzeit technologisch mögliche, bodengestützte, Reaktionszeit von Satelliten bestenfalls im Bereich von einigen Sekunden.

Diese Einschränkung stellt ein schweres Hindernis für neuartige Satellitenmissionen, bei denen insbesondere nichtdeterministische und kurzzeitige Phänomene (z.B. Blitze und Meteoreintritte in die Erdatmosphäre) Gegenstand der Beobachtungen sind, dar. Die langen Reaktionszeiten des konventionellen Satellitenbetriebs verhindern die Realisierung solcher Missionen, da die verzögerte Reaktion erst erfolgt, nachdem das zu beobachtende Ereignis bereits abgeschlossen ist.

Die vorliegende Dissertation zeigt eine Möglichkeit, das durch die langen Reaktionszeiten entstandene Problem zu lösen, auf. Im Zentrum des Lösungsansatzes steht dabei die Autonomie. Im Wesentlichen geht es dabei darum, den Satelliten mit der Fähigkeit auszustatten, sein Verhalten, d.h. die Folge seiner Tätigkeiten, eigenständig zu bestimmen bzw. zu ändern. Dadurch wird die direkte Abhängigkeit des Satelliten vom Operator bei Reaktionen aufgehoben. Im Grunde wird der Satellit in die Lage versetzt, sich selbst zu kommandieren.

Die Idee der Autonomie wurde im Rahmen der zugrunde liegenden Forschungsarbeiten umgesetzt. Das Ergebnis ist ein autonomes Planungssystem. Dabei handelt es sich um ein Softwaresystem, mit dem sich autonomes Verhalten im Satelliten realisieren lässt. Es kann an unterschiedliche Satellitenmissionen angepasst werden. Ferner deckt es verschiedene Aspekte des autonomen Satellitenbetriebs, angefangen bei der generellen Entscheidungsfindung der Tätigkeiten, über die zeitliche Ablaufplanung unter Einbeziehung von Randbedingungen (z.B. Ressourcen) bis hin zur eigentlichen Ausführung, d.h. Kommandierung, ab. Das Planungssystem kommt als Anwendung in ASAP, einer autonomen Sensorplattform, zum Einsatz. Es ist ein optisches System und dient der Detektion von kurzzeitigen Phänomenen und Ereignissen in der Erdatmosphäre.

Die Forschungsarbeiten an dem autonomen Planungssystem, an ASAP sowie an anderen zu diesen in Bezug stehenden Systemen wurden an der Professur für Raumfahrttechnik des Lehrstuhls Informatik VIII der Julius-Maximilians-Universität Würzburg durchgeführt.

Abstract

Satellite operation will change thoroughly in future. So far the currently performed conventional approach of controlling satellites is hitting its limitation by todays application. This is due to the fact that activities of the satellite are planned and controlled exclusively by ground infrastructure. In the worst case this circumstance prevents the exploitation of potential but so far unused opportunities. Thus the yield of satellites, may it be in the form of scientific research data or the commercialization of satellite services, is not optimized.

After all the cause of this problem can be tracked back to one crucial matter: Conventional satellites are not able to alter their behaviour, i.e. the order of their actions, themselves. Instead schedules are created by ground staff – mainly operators - utilizing specialized planning software. The output is then transformed into command sequences and uploaded to the dedicated satellite via ground stations. On-board the commands are solely checked, interpreted and strictly executed. The flow is linear. Situational changes, like in the code execution of software programs via control constructs, e.g. loops and branches, are typically not present. Thus the operator is the only instance which is able to change the behaviour of the satellite via command upload. Therefore a direct radio link between satellite and ground station is required. Reaction times are hereby restricted. In the best case, means when the satellite is inside the area of effect, the limitations are to a few seconds. Outside the contact window, the time bound may increase from minutes to hours. The exact timing are dependant from the orbit of the satellite and its position on it. The signal flow of the radio links adds additional reaction time from a few seconds in near earth up to some minutes in interplanetary space due to the vast distances. In sum the best achievable ground based reaction time lies in the area of some seconds.

This restriction is a severe handicap for novel satellite missions with focus on non-deterministic and short-time phenomena, e.g. lightning and meteor entries into Earth atmosphere. The long reaction times of conventional satellite operations prevent the realization of such missions. This is due to the fact that delayed reactions take place after the event to observe has finished.

This dissertation shows a possibility to solve the problem caused by long reaction times. Autonomy lies in the centre of the main approach. The key is to augment the satellite with the ability to alter its behaviour, i.e. the sequence of its actions, and to deliberate about it. Thus, the direct reaction dependency of the satellite from operators will be lifted. In principle the satellite will be able to command itself.

The herein idea of autonomy is based on research work, which provides the context for design and implementation. The output is an autonomous planning system. It's a software system which enables a satellite to behave autonomously and can be adapted to different types of satellite missions. Additionally, it covers different aspects of autonomous satellite operation, starting with general decision making of activities, going over to time scheduling inclusive constraint consideration, e.g. resources, and finishing at last with the actual execution, i.e. commanding. The autonomous planning system runs as one application of ASAP, an autonomous sensor platform. It is an optical system with the purpose to detect short-time phenomena and events in Earth atmosphere.

The research work for the autonomous planning system, for ASAP and for other related systems has been executed at the professorship for space technology which is part of the department for computer science VIII at the Julius-Maximilians-Universität Würzburg.

Inhaltsübersicht

1 Einführung ... 1
1.1 Strukturierung der Dissertation ... 1
1.2 Hintergrund / Projektumfeld ... 2
1.3 Problem/Fragestellung und deren Relevanz ... 5
1.4 Vision ... 7
1.5 Kontext ... 9
1.6 Zielsetzung und Abgrenzung ... 10
2 Fachbegriffe und benötigtes Grundwissen ... 11
2.1 Autonomie ... 11
2.2 Raumfahrttechnik ... 11
2.3 Ablaufplanung ... 21
2.4 Modellierungssprache SysML ... 25
2.5 RODOS ... 25
3 Beispiele autonomer Satelliten ... 28
3.1 Solar Maximum Mission ... 28
3.2 Earth Observing One ... 28
3.3 PROBA ... 29
3.4 SWIFT ... 29
3.5 XSS 11 ... 30
3.6 PLEIADES ... 30
4 Stand der Technik und der Forschung ... 31
4.1 Allgemeine Definitionen von autonomen Planungssystemen ... 31
4.2 Softwareagenten ... 32
4.3 Framesworks: ASPEN und CASPER ... 33
4.4 Autonomous Sciencecraft Experiment ... 34
4.5 Reaktive/Deliberative - Architektur ... 36
4.6 Ablaufplanungsalgorithmen für Erdbeobachtungen ... 36
4.7 ADEPT - Framework ... 38
4.8 AGATA ... 40
4.9 APSI ... 43
4.10 VAMOS ... 44
4.11 NEAT ... 45
4.12 Auswertung und Abgrenzung ... 47
5 Methode ... 50
6 Konzept ... 54
7 Realisierung ... 60

7.1 Referenzobjekte und Referenzszenarien ... 61
7.2 APS – Autonomes Planungssystem ... 66
7.3 ASAP ... 114
7.4 ATUS – Testumgebungssystem ... 143
7.5 Modellbasierte Systementwicklung ... 162
8 Ergebnisse ... 169
8.1 Entwurfsmethode ... 169
8.2 Autonome Planung ... 169
8.3 Geplanter Betrieb von ASAP in SONATE ... 175
8.4 Validierung der Anforderungen ... 178
9 Zusammenfassung ... 180
10 Ausblick ... 182
11 Anhang ... 183
12 Literaturverzeichnis ... 184

Inhaltsverzeichnis

1 Einführung ... 1
1.1 Strukturierung der Dissertation ... 1
1.2 Hintergrund / Projektumfeld ... 2
1.2.1 ASAP ... 2
1.2.2 ATUS ... 3
1.2.3 STELLA ... 3
1.2.4 RODOS ... 4
1.2.5 ADIA ... 4
1.2.6 SONATE ... 5
1.3 Problem/Fragestellung und deren Relevanz ... 5
1.4 Vision ... 7
1.5 Kontext ... 9
1.6 Zielsetzung und Abgrenzung ... 10
2 Fachbegriffe und benötigtes Grundwissen ... 11
2.1 Autonomie ... 11
2.2 Raumfahrttechnik ... 11
2.2.1 Raumfahrtmissionen und Raumfahrtsegmente ... 11
2.2.2 Subsysteme des Satelliten ... 13
2.2.3 Datenverarbeitung am Boden (Missionsbetrieb) ... 14
2.2.4 Datenverarbeitung im Satelliten ... 15
2.2.4.1 Bordcomputer ... 16
2.2.4.2 Kommandierung und Steuerung des Satelliten ... 17
2.2.4.3 Telemetrierung und Überwachung des Satelliten ... 19
2.2.5 Datenübertragung zwischen Satelliten und Bodenstation ... 20
2.3 Ablaufplanung ... 21
2.4 Modellierungssprache SysML ... 25
2.5 RODOS ... 25
3 Beispiele autonomer Satelliten ... 28
3.1 Solar Maximum Mission ... 28
3.2 Earth Observing One ... 28
3.3 PROBA ... 29
3.4 SWIFT ... 29
3.5 XSS 11 ... 30
3.6 PLEIADES ... 30
4 Stand der Technik und der Forschung ... 31
4.1 Allgemeine Definitionen von autonomen Planungssystemen ... 31

4.2 Softwareagenten ... 32
4.3 Framesworks: ASPEN und CASPER ... 33
4.4 Autonomous Sciencecraft Experiment ... 34
4.5 Reaktive/Deliberative - Architektur ... 36
4.6 Ablaufplanungsalgorithmen für Erdbeobachtungen ... 36
4.7 ADEPT - Framework ... 38
4.8 AGATA ... 40
4.9 APSI ... 43
4.10 VAMOS ... 44
4.11 NEAT ... 45
4.12 Auswertung und Abgrenzung ... 47
5 Methode ... 50
6 Konzept ... 54
7 Realisierung ... 60
7.1 Referenzobjekte und Referenzszenarien ... 61
7.2 APS – Autonomes Planungssystem ... 66
7.2.1 Zielstellung ... 66
7.2.2 Stakeholder ... 66
7.2.3 Anforderungen ... 67
7.2.4 Systemkontext ... 69
7.2.5 Systemsynthese ... 71
7.2.5.1 Hauptsteuerung ... 74
7.2.5.2 Datenverwaltung ... 76
7.2.5.3 Ereignisverwaltung ... 78
7.2.5.4 Entscheidungsfindung ... 79
7.2.5.5 Ablaufplanung ... 82
7.2.5.6 Planausführung ... 99
7.2.6 Besondere Aspekte ... 110
7.2.6.1 Softwaredienste ... 110
7.2.6.2 Sicherheit des Raumsegments ... 110
7.2.6.3 Kommunikation mit Operator ... 112
7.3 ASAP ... 114
7.3.1 Zielstellung ... 114
7.3.2 Stakeholder ... 115
7.3.3 Anforderungen ... 116
7.3.4 Systemkontext ... 121
7.3.5 Systemsynthese ... 122
7.3.5.1 Hardware ... 123
7.3.5.2 Schaltkreise ... 128

7.3.5.3 Software ... 135
7.3.5.3.1 Basis ... 135
7.3.5.3.2 Autonomes Bildsensorsystem ... 136
7.3.5.3.3 Autonomes Planungssystem ... 138
7.4 ATUS – Testumgebungssystem ... 143
7.4.1 Zielstellung ... 143
7.4.2 Stakeholder ... 144
7.4.3 Anforderungen ... 145
7.4.4 Systemkontext ... 149
7.4.5 Systemsynthese ... 150
7.4.5.1 Hardware (HIL) ... 152
7.4.5.2 Software (SIL) ... 153
7.4.5.2.1 Anwender-, Daten- und Persistenzschicht ... 154
7.4.5.2.2 Umgebungssimulation und 3D – Visualisierung ... 155
7.4.5.2.3 Lageregelung ... 158
7.5 Modellbasierte Systementwicklung ... 162
7.5.1 Modellierung des Systems ... 163
7.5.2 Automatische Quellcodegenerierung ... 166
7.5.3 Automatische Dokumentationsgenerierung ... 167
8 Ergebnisse ... 169
8.1 Entwurfsmethode ... 169
8.2 Autonome Planung ... 169
8.2.1 Deterministische Entscheidungsumsetzung ohne Optimierung ... 170
8.2.2 Anschließende Optimierung (nach Entscheidungsumsetzung) ... 171
8.2.3 Gleichzeitige Entscheidungsumsetzung und Optimierung ... 172
8.2.4 Reparatur ungültiger Pläne ... 173
8.2.5 Gleichzeitige Planung und Ausführung von Plänen (Fixierung) ... 173
8.2.6 Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität ... 174
8.3 Geplanter Betrieb von ASAP in SONATE ... 175
8.4 Validierung der Anforderungen ... 178
9 Zusammenfassung ... 180
10 Ausblick ... 182
11 Anhang ... 183
12 Literaturverzeichnis ... 184

Abkürzungsverzeichnis

ACS	Attitude Control System (dt. Lageregelungssystem)
APS	Autonomes Planungssystem
BMWi	Bundesministerium für Wirtschaft und Energie
CCSDS	Consultative Committee for Space Data Systems
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DP	Data Pool
ECEF	Earth Centered Earth Fixed
ECSS	European Cooperation for Space Standardization
ESA	European Space Agency
FKZ	Förderkennzeichen
FPGA	Field Programmable Gate Array
GSOC	German Space Operations Center (dt. Deutsches Raumfahrt-Kontrollzentrum)
HAL	Hardware Abstraction Layer
HIL	Hardware-In-The-Loop
LEOP	Launch and Eartly Orbit Phase
MBSE	Model Based System Engineering
OBC	Onboard Computer
OBCP	Onboard Control Procedure
OBDH	Onboard Data Handling
OSI/ISO	Open Systems Interconnection / International Organization for Standardization
PDH	Payload Data Handling
PUS	Packet Utilization Standard
RODOS	Realtime Onboard Dependable Operating System
RTOS	Realtime Operating System
SETI	Search for Extraterrestrial Intelligence
SIL	Software-In-The-Loop
SysML	System Modelling Language
UML	Unified Modelling Language
TC	Telecommand
TM	Telemetry

Abbildungsverzeichnis

Abbildung 1-1: ASAP ... 2
Abbildung 1-2: ATUS ... 3
Abbildung 1-3: STELLA ... 3
Abbildung 1-4: ADIA (Logo) ... 4
Abbildung 1-5: SONATE ... 5
Abbildung 1-6: konventioneller Satellitenbetrieb ... 7
Abbildung 1-7: autonomer Satellitenbetrieb (Vision) ... 8
Abbildung 1-8: Funktionalitäten der autonomen Planung ... 9
Abbildung 2-1: Raumfahrtmission und Raumfahrtsegmente ... 12
Abbildung 2-2: Sub-Systeme des Satelliten-Bus ... 13
Abbildung 2-3: Aufgaben und Abläufe des Missionsbetriebes ... 14
Abbildung 2-4: Software-Komponenten eines Bordcomputers [10, p. S.111] ... 16
Abbildung 2-5: Grundlegende Arbeitsschritte bei der Kommandierung ... 18
Abbildung 2-6: Grundlegende Arbeitsschritte bei der Erstellung von Telemetrie ... 19
Abbildung 2-7: Datenübertragung nach CCSDS [18, p. S. 2.4] ... 21
Abbildung 2-8: Prozess zur Ablaufplanung ... 22
Abbildung 2-9: Beispiele einer nichtpräemptiven und einer präemptiven geplanten Aktivität ... 24
Abbildung 2-10: RODOS und seine Komponenten (Quelle: DLR) ... 26
Abbildung 2-11: Datenübertragung mit Hilfe der Middleware (Quelle: DLR) ... 27
Abbildung 3-1: SMM Satellit (Quelle: NASA) ... 28
Abbildung 3-2: EO-1 Satellit (Quelle: NASA) ... 28
Abbildung 3-3: PROBA Satellit (Quelle: ESA) ... 29
Abbildung 3-4: SWIFT Satellit (Quelle: NASA) ... 29
Abbildung 3-5: XSS 11 Satellit (Quelle: ww.space.com) ... 30
Abbildung 3-6: PLEIADES Satellit (Quelle: CNES) ... 30
Abbildung 4-1: Aufgaben eines autonomen Planungssystems ... 31
Abbildung 4-2: Aktionen beim Planen ... 31
Abbildung 4-3: Kontinuierliches Planen und inkrementelles Erweitern des Plans (Quelle: [29]) ... 33
Abbildung 4-4: Architektur des Software-Agenten für das ASE (Quelle: [33]) ... 35
Abbildung 4-5: Reaktive/Deliverative – Architektur (Quelle: [38]) ... 36
Abbildung 4-6: Beispiel einer geplanten Spur einer Kamera an Bord eines Satelliten (Quelle: [32]) ... 37
Abbildung 4-7: Hierarchien im ADEPT – Framework (Quelle: [41]) ... 38
Abbildung 4-8: funktionale Zerlegung im ADEPT – Framework (Quelle: [41]) ... 39
Abbildung 4-9: Aufbau eines generischen Kontroll-Moduls (Quelle: [44]) ... 41
Abbildung 4-10: Steuerung eines Systems mit Kontroll-Modulen (Quelle: [44]) ... 42
Abbildung 4-11: Zeitspannen des Planungsalgorithmus (Quelle: [44]) ... 42
Abbildung 4-12: APSI Timeline Representation Framework (Quelle: [46]) ... 43
Abbildung 4-13: Aufbau eines kompletten Systems basierend auf APSI (Quelle: [46]) ... 44
Abbildung 4-14: OBoTiS - Abweichung eines Ressourcenverlaufs, dadurch Erweiterung der Kommandosequenz möglich (Quelle: [47]) ... 45
Abbildung 4-15: OBoTiS – Aktivierung einer Erweiterung der Kommandosequenz (Quelle: [47]) ... 45
Abbildung 4-16: OBETTE – Erstellung einer Erweiterung (Quelle: [47]) ... 45
Abbildung 4-17: NEAT – Aufbau (Quelle: [50]) ... 46
Abbildung 4-18: Abbildung 4-19: NEAT – genetischer Algorithmus (Quelle: [50]) ... 46
Abbildung 4-20: NEAT – Neuronales Netzwerk zur Korrektur der Res.-Propagation (Quelle: [51]) ... 47
Abbildung 5-1: Prozessschritte der verwendeten Entwurfsmethode ... 51
Abbildung 5-2: Prozessschritte während der Entwurfssynthese ... 52
Abbildung 6-1: Anwendungsfälle für den autonomen Betrieb eines Satelliten ... 54
Abbildung 6-2: In die Autonomie involvierte Teilnehmer/Komponenten ... 55
Abbildung 6-3: Anwendungsfälle des APS-Moduls ... 55

Abbildung 6-4: Darstellung der Interaktion zum autonomen Planen als Aktivität 56
Abbildung 7-1: Hierarchische Übersicht der Systeme 60
Abbildung 7-2: Ablauf für kurzfristige Beobachtungen 62
Abbildung 7-3: Darstellung aus der Sicht des Sensors 62
Abbildung 7-4: Detektierung 63
Abbildung 7-5: Aufnahme und Nachverfolgung 63
Abbildung 7-6: Abschluss 63
Abbildung 7-7: Detektierung, Nachverfolgung und Übertragung 64
Abbildung 7-8: Wiederaufnahme, Nachverfolgung und Übertragung 64
Abbildung 7-9: Aufnahme von Gebieten auf der Erde 64
Abbildung 7-10: Beobachtung und Nachverfolgung 65
Abbildung 7-11: Nachverfolgung 65
Abbildung 7-12: Nachverfolgung und Beobachtung 65
Abbildung 7-13: Stakeholders von APS 66
Abbildung 7-14: Systemkontext des APS 70
Abbildung 7-15: abstrahierter Systemkontext des APS mit der Ground-Anwendung 71
Abbildung 7-16: funktionales Verhalten der autonomen Planung (Übersicht) 71
Abbildung 7-17: interne Struktur des APS 73
Abbildung 7-18: Abläufe in der Hauptsteuerung 74
Abbildung 7-19: Zustände der Hauptsteuerung 75
Abbildung 7-20: Extrahierung ausgewählter Daten 76
Abbildung 7-21: Interaktionen zur Datenverwaltung beim autonomen Planungssystem 77
Abbildung 7-22: Passive Komponenten der Datenverwaltung im Daten-Pool 77
Abbildung 7-23: Interaktionen zur Ereignisverwaltung beim autonomen Planungssystem 78
Abbildung 7-24: Blöcke des Datenpools für die Eventverwaltung 79
Abbildung 7-25: Interaktionen zur Entscheidungsfindung (Übersicht) 80
Abbildung 7-26: Beziehung zwischen Entscheider und Prüfer 80
Abbildung 7-27: Blöcke des Prüfers APS Deliberator RuleBased 81
Abbildung 7-28: Interaktionen zur Ablaufplanung (Übersicht) 82
Abbildung 7-29: Schematische Darstellung eines Plans 83
Abbildung 7-30: Zustandsmaschine für Aktivitäten (vereinfachte Darstellung) 83
Abbildung 7-31: An der Ablaufplanung beteiligte Blöcke des Daten-Pools 84
Abbildung 7-32: Definition eines Profils 85
Abbildung 7-33: Anwendung eines Profils auf eine Ressource 85
Abbildung 7-34: Interaktionen zur Ablaufplanung (Datenfluss) 86
Abbildung 7-35: Interaktionen zur Ablaufplanung (Kontrollfluss) 88
Abbildung 7-36: Funktionalitäten der Ablaufplanung und deren Zusammensetzung 89
Abbildung 7-37: Zuordnung der Aktion j zu JK 93
Abbildung 7-38: Auflösen einer Abwägung bei der Ablaufplanung 94
Abbildung 7-39: Fixieren der Ordnung von Aktionen (Items) 96
Abbildung 7-40: Such-Baum (Lösungsraum) der Bindung 97
Abbildung 7-41: Zustandsmaschine für Aktivitäten zur Unterstützung der Ablaufplanung 98
Abbildung 7-42: Interaktionen zur Planausführung (Übersicht) 99
Abbildung 7-43: An der Planausführung beteiligte Blöcke des Daten-Pools 102
Abbildung 7-44: Darstellung der Horizonte und der zeitlichen Referenzpunkte (Beispiel) 104
Abbildung 7-45: Zustandsmaschine für Kommandos zur Unterstützung der Planausführung 106
Abbildung 7-46: Zustandsmaschine für Aktionen zur Unterstützung der Planausführung 107
Abbildung 7-47: Zustandsmaschine für Aktivitäten zur Unterstützung der Planausführung 109
Abbildung 7-48: struktureller Aufbau von Softwarediensten 110
Abbildung 7-49: Einbindung der FDIR-Anwendung in den Planungsprozess 111
Abbildung 7-50: Kommunikation zwischen Operator und APS (abstrakte Darstellung) 112
Abbildung 7-51: Kommunikation zwischen Operator und APS (detaillierte Darstellung) 113
Abbildung 7-52: Interner Aufbau einer COM-Anwendung 113
Abbildung 7-53: ASAP (Urheber: Oleskii Balagurin) 114

Abbildung 7-54: Stakeholders von ASAP ... 115
Abbildung 7-55: Systemkontext von ASAP ... 121
Abbildung 7-56: wesentliche funktionale Aspekte des ASAP Systems ... 122
Abbildung 7-57: Interne Struktur von ASAP Hardware ... 123
Abbildung 7-58: ASAP Hardware und Schnittstelle (Urheber: Oleskii Balagurin) ... 124
Abbildung 7-59: Konzept der Optik von ASAP (Urheber: Oleskii Balagurin) ... 124
Abbildung 7-60: Sensorplatine 1 ... 125
Abbildung 7-61: Sensorplatine 2 ... 125
Abbildung 7-62: Interne Struktur der Hauptplatine von ASAP ... 126
Abbildung 7-63: ASAP Hauptplatine ... 127
Abbildung 7-64: Verarbeitung der Bilddaten als Hauptfunktionalität von ASAP ... 129
Abbildung 7-65: Konfiguration des FPGAs ... 130
Abbildung 7-66: Prozessor, Speicher und Busse im FPGA ... 131
Abbildung 7-67: Arten von Speicher ... 132
Abbildung 7-68: externe Kommunikation ... 133
Abbildung 7-69: interne Kommunikation ... 134
Abbildung 7-70: Ansteuerung der Bildsensoren ... 134
Abbildung 7-71: Gemeinsame Funktionalitäten der ASAP Anwendungen ... 135
Abbildung 7-72: Funktionalitäten der Basis-Anwendung ... 136
Abbildung 7-73: Abläufe bei der Aufbereitung der Meta-Daten ... 136
Abbildung 7-74: Bestimmung der Richtung des Zielobjekts ... 138
Abbildung 7-75: Profil für exklusiven Zugriff auf eine Ressource ... 139
Abbildung 7-76: Profil für eine statische Auswirkung auf eine Ressource ... 140
Abbildung 7-77: Profil für integrierende Auswirkung auf eine Ressource ... 140
Abbildung 7-78: Sichtbarkeitsbereich von Objekten auf der Erde ... 141
Abbildung 7-79: Sichtbarkeitsbereich von Objekten auf Umlaufbahnen ... 141
Abbildung 7-80: Hardware-Testumgebung (HIL/ Aufbau) ... 143
Abbildung 7-81: Software-Testumgebung (SIL / 3D-Visualisierung) ... 143
Abbildung 7-82: Verschiedene Varianten von Testaufbauten ... 144
Abbildung 7-83: Stakeholder von ATUS ... 144
Abbildung 7-84: Systemkontext von ATUS ... 149
Abbildung 7-85: funktionales Verhalten des HILs von ATUS ... 150
Abbildung 7-86: funktionales Verhalten des SILs von ATUS ... 151
Abbildung 7-87: Gehäuse des HIL (Hardware) – Ansicht auf das Frontpanel ... 152
Abbildung 7-88: Platine des HIL (Hardware) ... 153
Abbildung 7-89: graphische Benutzeroberfläche ... 154
Abbildung 7-90: 3D - Visualisierung ... 155
Abbildung 7-91: statische Struktur der Raumsimulation ... 156
Abbildung 7-92: Transformationen zwischen Koordinatensystemen ... 158
Abbildung 7-93: Horizont-Modus (Außen-Ansicht) ... 159
Abbildung 7-94: Horizont-Modus (FOV-Ansicht) ... 159
Abbildung 7-95: vor der Ausrichtung ... 160
Abbildung 7-96: nach der Ausrichtung ... 160
Abbildung 7-97: Übersicht der modellbasierten Systementwicklung ... 162
Abbildung 7-98: Datenverwaltung ... 163
Abbildung 7-99: Datenmodell für ASAP und SONATE (Schema Top) ... 165
Abbildung 7-100: Datentransformation (Dokumentation) ... 167
Abbildung 8-1: Deterministische Entscheidungsumsetzung ohne Optimierung (Messreihe A Index 4) ... 171
Abbildung 8-2: An vorherige Entscheidungsumsetzung anschließende Optimierung (Messreihe B Index 13) ... 171
Abbildung 8-3: Gleichzeitige Entscheidungsumsetzung und Optimierung (Messreihe C Index 15) ... 172
Abbildung 8-4: Reparatur ungültiger Pläne (Messreihe D16) ... 173
Abbildung 8-5: Gleichzeitige Planung und Ausführung von Plänen (Messreihe E10) ... 174

Abbildung 8-6: Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität (Messreihe F19) 175
Abbildung 8-7: Missionsbetrieb von SONATE - Definition von Telekommandos und Telemetrie 176
Abbildung 8-8: Missionsbetrieb von SONATE – Datenfluss von Telemetrie und Telekommando 177

Tabellenverzeichnis

Tabelle 1-1: Eingruppierung der Satelliten hinsichtlich Masse, Volumen und el. Leistung 10
Tabelle 2-1: Der Aufbau eines RODOS-Frames 27
Tabelle 4-1: Eigenschaften von Software-Agenten 32
Tabelle 4-2: Gegenüberstellung autonomer Planungssysteme 48
Tabelle 7-1: Nutzeranforderungen an das APS 67
Tabelle 7-2: Systemanforderungen (funktionale Anforderungen) an das APS 69
Tabelle 7-3: Systemanforderungen (Qualitätsanforderungen) an das APS 69
Tabelle 7-4: Systemanforderungen (Randbedingungen) an das APS 69
Tabelle 7-5: Strategien der Ablaufplanung 98
Tabelle 7-6: Der strukturelle Aufbau eines Kommandos 100
Tabelle 7-7: Definition (Beispiel) 103
Tabelle 7-8: Plan (Beispiel) 103
Tabelle 7-9: Horizonte (Beispiel) 104
Tabelle 7-10: Erstellte Kommandofolge in Abhängigkeit der Zeit (Beispiel) 105
Tabelle 7-11: Nutzeranforderungen an das ASAP System 118
Tabelle 7-12: Systemanforderungen (funktionale Anforderungen) an das ASAP System 120
Tabelle 7-13: Systemanforderungen (Qualitätsanforderungen) an das ASAP System 120
Tabelle 7-14: Systemanforderungen (Randbedingungen) an das ASAP System 120
Tabelle 7-15: Struktur der Meta-Daten (Endformat) 137
Tabelle 7-16: Ressourcen der autonomen Planung 139
Tabelle 7-17: Aktivitäten der autonomen Planung 139
Tabelle 7-18: Nutzeranforderungen an das ATUS - System 145
Tabelle 7-19: Systemanforderungen (funktionale Anforderungen) an das ATUS System 148
Tabelle 7-20: Systemanforderungen (Qualitätsanforderungen) an das ATUS System 149
Tabelle 7-21: Systemanforderungen (Randbedingungen) an das ATUS System 149
Tabelle 7-22: Realisierungsgrad der automatischen Codegenerierung 166
Tabelle 8-1: Deterministische Entscheidungsumsetzung ohne Optimierung (Messreihe A) 170
Tabelle 8-2: An vorherige Entscheidungsumsetzung anschließende Optimierung (Messreihe B) 172
Tabelle 8-3: Gleichzeitige Entscheidungsumsetzung und Optimierung (Messreihe C) 172
Tabelle 8-4: Reparatur ungültiger Pläne (Messreihe D16) 173
Tabelle 8-5: Gleichzeitige Planung und Ausführung von Plänen (Messreihe E10) 174
Tabelle 8-6: Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität (Messreihe F) 175
Tabelle 8-7: Validierung der Anforderungen an das APS 178
Tabelle 8-8: Validierung der Anforderungen an ASAP 178
Tabelle 8-9: Validierung der Anforderungen an ATUS 179

1 Einführung

1.1 Strukturierung der Dissertation

Die vorliegende Dissertation ist didaktisch aufgebaut. Sie ist in Kapitel (1. Ebene) und Abschnitte (2. - *.Ebene) untergliedert.

Das vorliegende Kapitel 1 „Einführung“ beginnt mit der Beschreibung des Strukturaufbaus (Abschnitt 1.1) und wird von einer Erläuterung des Hintergrundes bzw. des Projektumfeldes des Autors (Abschnitt 1.2) gefolgt. Dieses zeigt auf, aus welchen Forschungsarbeiten die Dissertation entstanden ist. Als nächstes (Abschnitt 1.3) wird das eigentliche Problem erläutert und als Fragestellung formuliert. Im darauffolgenden Abschnitt 1.4 wird zu dieser ein Lösungsansatz in Form einer Vision skizziert. Zur Orientierung wird im nächsten Schritt (Abschnitt 1.5) der wissenschaftliche Kontext dieser Arbeit erfasst. Am Ende der Einleitung (Abschnitt 1.6) wird die Zielsetzung aufgezeigt. Hierbei wird der Anspruch der vorliegenden Dissertation als wissenschaftliche Arbeit und als Beitrag zur Forschung erhoben. Zur Verdeutlichung wird dabei die wissenschaftliche Arbeit thematisch abgegrenzt.

Es wird davon ausgegangen, dass der Leser ein(e) grundlegende(s) natur- oder ingenieurwissenschaftliche(s) Ausbildung/Studium abgeschlossen hat. Das in der Dissertation behandelte Thema ist jedoch der Raumfahrttechnik, einem technischen Fachbereich, zuzuordnen. Deshalb werden im Kapitel 2 zunächst nicht generell als bekannt vorausgesetzte Fachbegriffe erläutert und benötigtes fachspezifisches Grundwissen ergänzend vermittelt. Beides ist für das Verständnis der vorliegenden Arbeit unbedingt notwendig. Zuerst (Abschnitt 2.1) wird dabei der Begriff der Autonomie im technischen Kontext allgemein erklärt. Anschließend (Abschnitt 2.2) wird auf besonders wichtige Aspekte der Raumfahrttechnik, zum Beispiel Subsysteme, Datenverarbeitung oder Missionsbetrieb, eingegangen. Bei letztgenanntem ist die Ablaufplanung von besonderer Bedeutung. Diese wird daher in einem eigenen Abschnitt (Abschnitt 2.3) erläutert. Generell wurden die in der Dissertation vorgestellten Systeme mit Hilfe der Modellierungsprache SysML entworfen. Fast alle dargestellten Diagramme sind in dieser Beschreibungssprache verfasst. Aus diesem Grund wird in einem gesonderten Abschnitt auf diesen Formalismus eingegangen (Abschnitt 2.4). Das Kapitel schließt mit einer kurzen Einführung in RODOS (Abschnitt 2.5) ab. Dies ist ein Betriebssystem, auf dem die in der Dissertation vorgestellten Anwendungen aufbauen.

Das nachfolgende Kapitel 3 soll dem Leser einen Eindruck davon vermitteln, inwieweit Autonomie schon heute in der Raumfahrttechnik eingesetzt wird, und welche Möglichkeiten sie beim Satellitenbetrieb eröffnet. Zu diesem Zweck werden ausgewählte Raumfahrtmissionen/Satelliten vorgestellt.

Kapitel 4 fasst die für eine wissenschaftliche Arbeit unerlässlichen Recherchen zu einem aktuellen Stand zur Forschung und Technik zusammen. Jeder Unterabschnitt widmet sich einem besonderen Ansatz. Am Ende des Kapitels (Abschnitt 4.12) werden die Ansätze verglichen und klassifiziert.

Mit dem Kapitel 5 beginnt die Beschreibung des im Rahmen der Forschungsarbeiten des Autors entwickelten autonomen Systems, das den eigentlichen Beitrag zur Forschung darstellt. Der Entwurf erfolgte anhand einer wissenschaftlichen Methode. Diese ist Gegenstand des Kapitels.

In Kapitel 6 wird ein erstes Konzept vorgestellt, das beschreibt, wie die Fragestellung bzw. das Problem der autonomen Planung gelöst werden kann.

Die eigentliche Ausarbeitung bzw. Realisierung folgt im Kapitel 7. Der Entwurf orientiert sich dabei an sogenannten Referenzszenarien (Abschnitt 7.1). Diese beschreiben bzw. veranschaulichen die Ziele, die mit Hilfe der Autonomie realisiert werden sollen. Die nachfolgenden Unterabschnitte (Abschnitt 7.2 - 7.4) behandeln die jeweiligen entworfenen (autonomen) Systeme im Detail. Der letzte Abschnitt in

diesem Kapitel geht speziell auf die verwendete Entwurfsmethode ein und zeigt, wie diese umgesetzt wurde.

Anschließend werden im Kapitel 8 die Ergebnisse vorgestellt. Dabei werden der aktuelle Stand bzw. die Fortschritte der im vorherigen Kapitel genannten Systeme erfasst, in einer Evaluierung ausgewertet und zusammengefasst.

Die nachfolgende Zusammenfassung (Kapitel 9) bildet ein Resümee der Inhalte der vorliegenden Dissertation.

Abschließend werden im Kapitel 10 mögliche Fortführungen zu den in der Dissertation behandelten Themen aufgezeigt.

Zuletzt werden die obligatorischen Inhalte „Anhang“ im Kapitel 11 und „Literaturverzeichnis“ in Kapitel 12 gelistet.

1.2 Hintergrund / Projektumfeld

Die vorliegende Dissertation entstand im Rahmen von Forschungsarbeiten an der Universität Würzburg am Lehrstuhl für Informatik VIII an der Professur für Raumfahrttechnik von Professor Hakan Kayal. Dieser Lehrstuhl beschäftigt sich insbesondere mit Thematiken im Bereich Luft- und Raumfahrtsysteme. Zu den Schwerpunkten der Professur gehören unter anderem die höhere Autonomie sowie die Erforschung unbekannter Himmelsphänomene. Die Dissertation basiert im Wesentlichen auf den Erkenntnissen aus den unten vorgestellten Projekten, die an diesem Lehrstuhl durchgeführt und im Rahmen von Vorhaben[1] des Deutschen Zentrums für Luft- und Raumfahrt (DLR) gefördert wurden.

Hinweis:

Die durch das Deutsche Zentrum für Luft- und Raumfahrt geförderten Vorhaben wurden mit Mitteln des Bundesministeriums für Wirtschaft und Technologie (BMWi) aufgrund eines Beschlusses des Deutschen Bundestages finanziert. Die den jeweiligen Vorhaben zugeordneten Förderkennzeichen (FKZ) können in den folgenden Unterabschnitten entnommen werden.

1.2.1 ASAP

Dieses Vorhaben [1], [2], [3], [4] bildet die Hauptgrundlage der vorliegenden Dissertation. Ziel war der Entwurf, die Implementierung und die Qualifikation eines technischen Systems zur Steigerung der autonomen Fähigkeiten von Kleinstsatelliten.

Abbildung 1-1: ASAP

Im Wesentlichen besteht ASAP aus zwei funktionellen Haupteinheiten: dem autonomen Sensor und dem autonomen Planungssystem. Der Sensor arbeitet optisch und hat die Aufgabe, bestimmte Ereignisse (z.B. Meteoreintritte) und Phänomene (z.B. Blitze) in der Erdatmosphäre bzw. im erdnahen Raum mittels Beobachtung zu detektieren und

[1] Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) verwendet generell den Begriff ‚Vorhaben‘ für seine durch ihn geförderten Projekte. In der Dissertation wird Vorhaben als Synonym für ein Projekt verwendet.

hieraus relevante beschreibende Daten (z.B. Art des Ereignisses) abzuleiten. Das autonome Planungssystem verwendet diese Daten, um daraus einen Plan zu bestimmen, der die zukünftigen Handlungen des Satelliten definiert. Diese Vorgänge finden kontinuierlich statt, sodass der Satellit stets in der Lage ist, auf neue Ereignisse, ohne direktes Einwirken des Operators, zu reagieren.

Das Vorhaben sieht die Herstellung eines Flugmodells von ASAP vor. Dieses ist für eine angedachte zukünftige Satellitenmission mit dem Namen ANEX-1 vorgesehen. Parallel laufen bereits Vorbereitungen, um Kernkomponenten von ASAP im Satelliten SONATE (siehe Abschnitt 0) einzusetzen und zu testen.

FKZ: 50RM1208

1.2.2 ATUS

Das Vorhaben ATUS stellt, thematisch betrachtet, eine Ergänzung zu ASAP dar. Hierbei stehen der Test und die Simulation von möglichen Satellitenmissionen in Form von Szenarien, bei denen autonome Funktionalitäten getestet werden sollen, im Vordergrund. Die Notwendigkeit für das Vorhaben ergab sich aus der Tatsache, dass zum Zeitpunkt der Fertigstellung von ASAP noch kein passendes Satellitensystem als Versuchsträger zur Verfügung stand. Um dennoch funktionale Tests an ASAP durchführen zu können, wird mit ATUS ein Testumgebungssystem entwickelt, das die Funktionalitäten sowohl eines Hardware-In-The-Loop (HIL) Systems als auch eines Software-In-The-Loop (SIL) Systems wahrnimmt. Des Weiteren wird im Rahmen des Vorhabens auch der Funktionsumfang von ASAP kontinuierlich erweitert.

Abbildung 1-2: ATUS

FKZ: 50RM1521

1.2.3 STELLA

STELLA [5] ist ein Sternsensor (eng. Star Tracker), der in Pico- und Nano- Satelliten eingesetzt werden kann. Er besitzt rudimentäre autonome Fähigkeiten. Seine Aufgabe besteht darin, mit Hilfe einer optischen Erfassung und einer anschließenden Datenverarbeitung die Ausrichtung des Satelliten im Weltall zu bestimmen. Diese Lageinformationen werden gewöhnlich an das Lageregelungs-Subsystem des Satelliten zur weiteren Verarbeitung übermittelt.

Abbildung 1-3: STELLA

Die Funktionsweise beruht auf einer kontinuierlichen Datenverarbeitung. Dabei werden vom Bildsensor autonom aufgenommene Sternbilder in mehreren Schritten prozessiert. Das Ziel besteht darin, die Lageinformation aus den Bildern zu extrahieren. Die nachfolgenden Arbeitsschritte werden kontinuierlich wiederholt:

1. Sternbilderfassung des Sternenhimmels im Sichtfeld der Sternkamera.
2. Erkennung von Sternen und anderen Objekten (z.B. Mond, Weltraumschrott, Pixelfehler) im Bild mit Hilfe spezieller Bildverarbeitungsalgorithmen.

3. Identifizierung der Sterne unter Einbeziehung der Abstände zwischen den im Bild erkannten Sternen und einem Vergleich mit einer internen Sternendatenbank im Speicher.
4. Berechnung der Lageinformation durch Relation der im Sichtfeld der Kamera identifizierten Sterne mit dem globalen Koordinatensystem (z.B. ECEF).

FKZ: 50RM0901

1.2.4 RODOS

RODOS [6] ist ein Software-Framework, das ursprünglich vom DLR (Standort Bremen) entwickelt wurde, und an deren Entwicklung Prof. Sergio Montenegro maßgeblich beteiligt war. Es ist modular und mehrschichtig aufgebaut. Die wichtigsten Komponenten sind das Echtzeit- Betriebssystem und die darauf aufsetzende Middleware. Der Einsatz von RODOS in HW/SW-Systemen hat den Vorteil, dass eine Abstrahierung von der Hardware stattfindet. Software-Anwendungen müssen lediglich die RODOS-Schnittstelle berücksichtigen und können wahlweise auf jeder beliebigen Plattform, auf der RODOS implementiert ist, eingesetzt werden. Ein Mechanismus basierend auf dem Publisher/Subscriber-Entwurfsmuster ermöglicht den Datenaustausch zwischen mehreren Software-Anwendungen.

Die Vorhaben ASAP (siehe Abschnitt 1.2.1) und SONATE (siehe Abschnitt 0) verwenden RODOS. Für deren Hardware-Plattformen mussten RODOS-Distributionen erstellt werden. Daneben kommt RODOS auch in anderen Vorhaben des Lehrstuhls für Informatik VIII zum Einsatz. Zu diesen gehören insbesondere VIDANA [7] (FKZ: 50RM1203) und YETE [8] (FKZ: 50RA1330), beide hochgradig skalierbare verteilte Datenmanagementsysteme für Kleinstsatelliten.

1.2.5 ADIA

ADIA [9] ist in erster Linie ein Diagnose-System, das als eine Komponente in einem Satelliten zum Einsatz kommen soll. Die Besonderheit des Systems liegt darin, dass es an Bord des Satelliten bereits aufgetretene Fehler nicht nur detektiert, sondern sie auch analysiert, um deren Ursachen zu bestimmen. Des Weiteren wird überprüft, ob sich Fehler in Zukunft anbahnen könnten. Dies geschieht mit Hilfe einer Trendanalyse. Zum einen wird durch die frühzeitige Erkennung mehr Zeit für die Fehlerbehebung bzw. -vermeidung gewonnen. Zum anderen können eventuell mögliche Fehler, noch bevor diese eintreten, eliminiert werden.

Abbildung 1-4: ADIA (Logo)

Von besonderem Interesse ist die Möglichkeit, Teile der Funktionalität von ADIA mit ASAP zu vereinen. Die Idee besteht darin, die Ergebnisse der autonomen Funktionen von ASAP mit Hilfe der Diagnosefunktionen von ADIA zu überprüfen, um festzustellen, ob die autonomen Handlungen den Satelliten in Zukunft in Gefahr bringen könnten.

FKZ: 50RM1231

1.2.6 SONATE

SONATE ist ein Satellit, der primär der Erprobung neuer Technologien dient. Im Vordergrund steht der autonome Betrieb von Satelliten. Deswegen gehören Kernkomponenten von ASAP und ADIA zu den Hauptbestandteilen des Satelliten. Die allumfassende Idee besteht darin, dass der Satellit seine Tätigkeiten selbst planen und ausführen soll. Des Weiteren sollen bei der Satellitenmission verschiedene Tests und Verifikationen durchgeführt werden. Dadurch soll sichergestellt werden, dass die beteiligten Komponenten auch unter realen Weltraum-bedingungen arbeiten können.

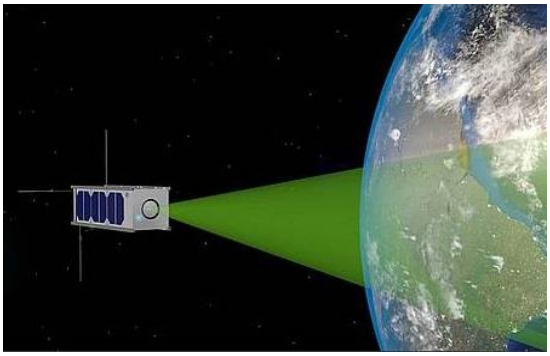


Abbildung 1-5: SONATE

Weitere untergeordnete Missionsziele sind:

- Ausbildung von Studierenden
- Erprobung von Lageregelungskomponenten
- Broadcasting von Bilddaten über Amateurfunkfrequenzen

Der Start des Satelliten ist für das Jahr 2019 geplant.

FKZ: 50RM1606

1.3 Problem/Fragestellung und deren Relevanz

Satelliten, die sich derzeit im Orbit befinden, werden nach dem heutigen Stand der Technik grundsätzlich vom Boden aus durch ein Netzwerk, bestehend aus Kontroll- und Bodenstationen, gesteuert. Dabei ist es möglich, die Befehle, auch Kommandos genannt, nicht nur zeitlich unmittelbar nach dem Empfang vom Satelliten, sondern erst mit einer bestimmten Verzögerung (time-tag), ausführen zu lassen. Zum Zeitpunkt der Ausführung kann sich der betroffene Satellit sogar außerhalb der Kommunikationsreichweiten der Bodenstationen befinden. Die Kommandos werden in einer speziell dafür vorgesehenen Liste zeitlich geordnet und verwaltet, sodass eine Befehlssequenz entsteht, die sequenziell vom Satelliten abgearbeitet werden kann. Dieser erlangt dadurch „niedere“ Autonomie [10, pp. 211-214], d. h. er wird teilweise und im geringen Umfang unabhängig von der ihn kontrollierenden Einrichtung.

Die Kommandosequenzen werden im normalen Betrieb am Boden und im Voraus vom Missions-Betriebsteam mit Hilfe spezieller Planungssoftware erstellt. In dieser Software kommen Ablauf-planungsalgorithmen zum Einsatz, die in ihren Berechnungen verschiedene Faktoren (z.B. Beschränkungen und Ressourcen des Satelliten) berücksichtigen. Die Kommandosequenzen werden somit kontinuierlich erstellt und zum Satelliten per Funk übertragen.

Diese konventionelle Vorgehensweise hat sich im operationellen Betrieb von Satelliten durchgesetzt und wird seit Jahrzehnten praktiziert. Der Grund liegt darin, dass Satelliten inkl. ihrer notwendigen Infrastruktur extrem wertvolle Güter darstellen. Die Kosten für Entwicklung und Betrieb betragen in der Regel mehrere Millionen Euro und können bei kommerziellen Großsatelliten sogar bis in die Milliarden gehen. Der Ausfall eines solchen Satelliten aufgrund eines fehlerhaften Betriebes, insbesondere der Kommandierung, stellt somit ein zu hohes Risiko dar. Deswegen hat sich der konventionelle Betrieb, bei dem die Operatoren die größtmögliche Kontrolle über den Satelliten vom Boden aus ausüben durchgesetzt. Allenfalls in der Fehlerbehandlung, also, wenn bereits schon ein Fehlerzustand im Satelliten eingetreten ist, wird üblicherweise ein geringes Maß an Autonomie eingesetzt, bei der der

Satellit automatisch ohne dem Eingreifen des Operators feste Prozeduren ausführt und so in einen sichereren Zustand (engl. fail-safe mode) übergeht.

Dennoch ist der konventionelle Betrieb mit einigen gravierenden Nachteilen verbunden:

1. Die Software-Systeme zur Erstellung der Kommandosequenzen müssen von einer hoch-professionellen Belegschaft bedient werden, die zudem in dem gesamten Planungsprozess des Satelliten eng eingebunden ist. Dies verursacht immense Personalkosten, die die gesamte Finanzierung der Satellitenmission belasten [11, pp. 45-52].
2. Der konventionelle Satellitenbetrieb schränkt sogar mögliche Anwendungen ein, und zwar insbesondere dann, wenn schnelle Reaktionen des Satelliten (im Bereich von Millisekunden) auf nicht deterministische Ereignisse erforderlich sind [1], [2]. Solche Anwendungen sind zum Beispiel die optische Nachverfolgung von am Satelliten vorbeiziehenden Objekten oder die Detektion von kurzzeitigen Phänomenen (z.B. Blitze, Sprites oder Meteoreintritte in die Erdatmosphäre). Der Grund hierfür liegt darin, dass eine Reaktion auf solche Ereignisse eine Anpassung der Kommandoliste im Satelliten erfordern würde. Diese setzt wiederrum ein Download von Information bezüglich des gerade geschehenden Ereignisses, gefolgt von einer Ablaufplanung zur Erstellung einer neuen Kommandosequenz, die eine entsprechende Reaktion auf das Ereignis beinhaltet, mit einem anschließenden Upload voraus. Dieser Vorgang, bestehend aus Download, Planung und Upload, verursacht eine nicht zu vernachlässigende Zeitverzögerung. Diese führt höchstwahrscheinlich dazu, dass das kurzzeitige Ereignis schon vorbei ist, bevor der Satellit auf dieses reagieren kann.
3. Es könnte der Fall eintreten, dass eine Information bezüglich eines Ereignisses aus diversen Gründen (z.B. Einschränkung der Kommunikationsbandbreite) vom Satelliten aus gar nicht zur Bodenstation übertragen wird. Dies wäre eine nicht wahrgenommene Gelegenheit (loss of opportunity), neue wissenschaftliche Erkenntnisse zu erlangen.
4. Die Ablaufplanungsalgorithmen müssen bei der Erstellung der Kommandosequenzen die zur Verfügung stehenden Ressourcen des Satelliten berücksichtigen. Dabei wird der Verlauf der Ressourcen für die Zukunft berechnet. Da die Berechnungen jedoch auf Modellen basieren, muss der tatsächliche Verlauf einer Ressource nicht unbedingt mit den berechneten Werten übereinstimmen. Dadurch kann es passieren, dass in Zukunft doch mehr Ressourcen zur Verfügung stehen, als ursprünglich geplant. Diese können jedoch erst benutzt werden, wenn der Satellit durch eine erneute Kommandierung neue Anweisungen erhält. Dazu ist ein neuer Kontakt zur Bodenstation nötig. Die Zeit bis dahin kann somit nicht optimal genutzt werden.

Die obigen genannten Nachteile lassen sich auf ein Problem zurückführen, das sich mit der folgenden Fragestellung beschreiben lässt:

Wie kann die Handlungsweise eines Satelliten dahingehend verbessert werden, dass er sowohl eigenständig auf nichtdeterministische Ereignisse reagieren, als auch neue deterministische Handlungen für sich festlegen kann? Die Eigenständigkeit soll in dem Sinn aufgefasst werden, dass die Abläufe im Satelliten im sehr hohen Grad nicht von der Bodenstation abhängig sind.

Generell ist das Problem auf die Abwesenheit „höherer“ autonomer Funktionalitäten des Satelliten zurück zu führen. Daher wird dieses Problem bzw. diese Fragestellung in der Dissertation als „Autonomes Planungsproblem“ bezeichnet.

Auf diese Fragestellung ist Prof. Kayal von der Universität Würzburg während seiner Forschung [11] gestoßen. Schwerpunktmäßig beschäftigt er sich mit folgenden Themen [12]:

- Klein- und Kleinstsatelliten für wissenschaftliche Anwendungen
- höhere Autonomie im Satellitenbetrieb
- Suche nach außerirdischer Intelligenz (SETI)
- Erforschung unbekannter Himmelsphänomene

Satelliten, deren funktionale Möglichkeiten nicht durch das autonome Planungsproblem eingeschränkt wären, könnten ein hilfreiches Mittel bei den oben genannten Forschungsthemen und anderen Weltraumanwendungen sein.

Das „höhere“ autonome Planungsproblem gewinnt stetig an Relevanz. Es verhindert die Anwendung neuartiger Applikationen, bei denen es vor allem, wie oben erwähnt wurde, auf schnelle (im Bereich von Millisekunden) und von der Bodenstation unabhängige Reaktionen des Satelliten, ankommt. Im Abschnitt 7 (Realisierung) werden einige mögliche Missionsszenarien und Ansätze, die aufzeigen, wie das Problem der autonomen Planung gelöst werden kann, präsentiert.

1.4 Vision

Der Kerngedanke zur Lösung des im obigen Abschnitt erläuterten autonomen Planungsproblems besteht darin, dem Satelliten eine Möglichkeit zu geben, seine interne Kommandoliste zu manipulieren. Dies ist der erste Schritt in Richtung Autonomie. Ferner soll die Autonomie als eine Erweiterung aufgefasst werden. Dies bedeutet, im Bedarfsfall soll ein konventioneller Betrieb des Satelliten möglich sein. Aus diesem Grund soll an dieser Stelle zuerst der konventionelle Satellitenbetrieb vereinfacht dargestellt werden (siehe Abbildung 1-6). Detailliertere Informationen zur Missionsplanung und zum Missionsbetrieb eines konventionellen Satelliten befinden sich im Abschnitt 2.2.3.

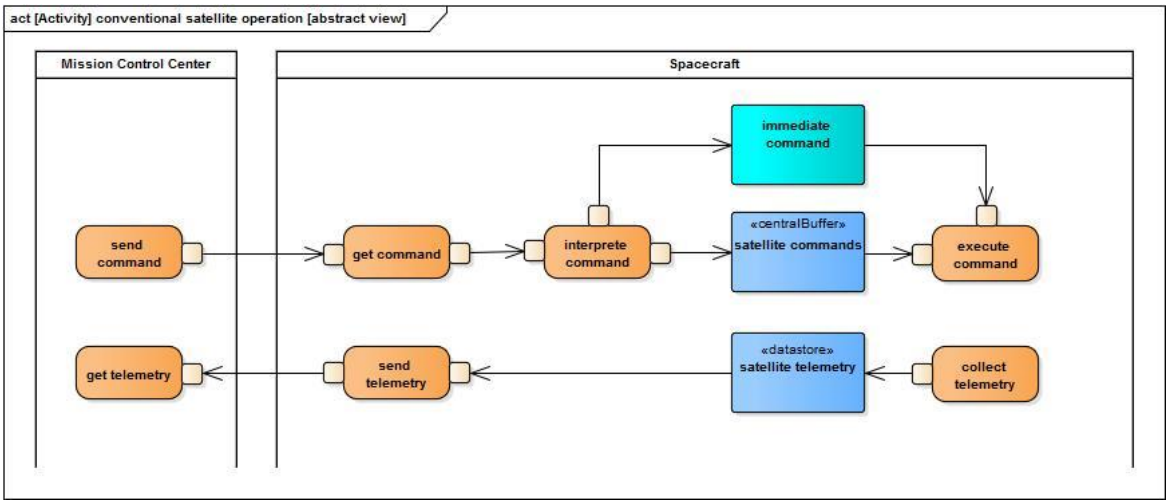


Abbildung 1-6: konventioneller Satellitenbetrieb

Bei der herkömmlichen bodengestützten Kommandierung gelangt ein Kommando über mehrere Zwischenstationen (z.B. Kontrollzentrum, Bodenstation, usw.) zuletzt zum Bordcomputer des Satelliten. Durch seine nachfolgende Interpretation wird das Kommando entweder vorläufig in einem Zwischenspeicher abgelegt, falls es mit einem Zeitstempel (time-tag) versehen ist, um später zu diesem spezifischen Zeitpunkt aus dem Speicher entnommen und ausgeführt zu werden, oder es handelt sich um ein Kommando, das sofort (immediate) ausgeführt werden soll. Des Weiteren werden im konventionellen Betrieb permanent Telemetriedaten (z.B. die Zustandsdaten des Satelliten, Nutzdaten der Instrumente) erfasst und zur Bodenkontrolle gesendet.

Eine direkte Manipulation der Kommandoliste im Bordcomputer ist normalerweise nicht vorgesehen, es sei denn, sie ist durch die Bodenstation veranlasst worden. Es gibt allerdings zwei Ausnahmen:

1. Üblicherweise werden während der LEOP-Phase (launch and early operation) einer Mission zur Vorbereitung und zur Inbetriebnahme des Satelliten automatisch ablaufende Prozeduren eigenständig durch ihn ausgeführt.

2. In besonders kritischen Ausnahmefällen, bei denen der Verlust des Satelliten auf Grund von Fehlerzuständen droht, sind eigenständige Operationen des Satelliten zum Übergang in einen „gesicherten Modus“ (safe-mode) vorgesehen (siehe Abschnitt 7.2.6.2)

Ansonsten erfolgen die Eingriffe in die Kommandoliste des Satelliten stets vom Boden aus. Genau an diesem Punkt setzt die Vision an (siehe Abbildung 1-7) und erweitert das konventionelle Konzept.

Der Satellit erhält eine Erweiterung. Über eine zu definierende Schnittstelle wird eine neue, intelligente Komponente - das autonome Planungssystem (siehe Realisierung in Abschnitt 7.2) - mit dem Bordcomputer des Satelliten verbunden. Über die Schnittstelle können Änderungen an der Kommandoliste vorgenommen und Telemetriedaten abgeglichen werden. Im konventionellen Satellitenbetrieb bleibt die Schnittstelle deaktiviert, sodass der autonome Zugriff auf die Kommandoliste verhindert wird und der Satellit dadurch lediglich herkömmlich vom Boden aus betrieben werden kann. Im autonomen Satellitenbetrieb, also mit aktivierter Schnittstelle, greift das autonome Planungssystem in die Steuerung des Satelliten ein. Drei Prozesse laufen dabei gleichzeitig ab: Der erste legt Handlungen des Satelliten in Form von Aktionen und Reaktionen fest. Dies kann auf der Basis zu verarbeitender Eingangsdaten erfolgen. Dazu gehören zum einen die externen Ereignisse deterministischer und nichtdeterministischer Art (siehe Szenarien in Abschnitt 7.1). Zum anderen können auch Telemetriedaten direkt ausgewertet werden. Zuletzt besteht noch die Möglichkeit, dass Handlungen via Kommandierung direkt vom Boden vorgegeben werden. Dieses Feature dient als Ersatz dafür, dass eine direkte Manipulation der Kommandoliste vom Boden aus nicht mehr möglich ist. In einem zweiten Prozess müssen die getroffenen Entscheidungen in einem Plan umgesetzt werden, aus dem schließlich in einem letzten Prozess Kommandos für die Bordsteuerung des Satelliten abgeleitet werden. Eine Synchronisation auf dem OBC bindet die autonom generierten Kommandos in die ausführende Kommandoliste ein.

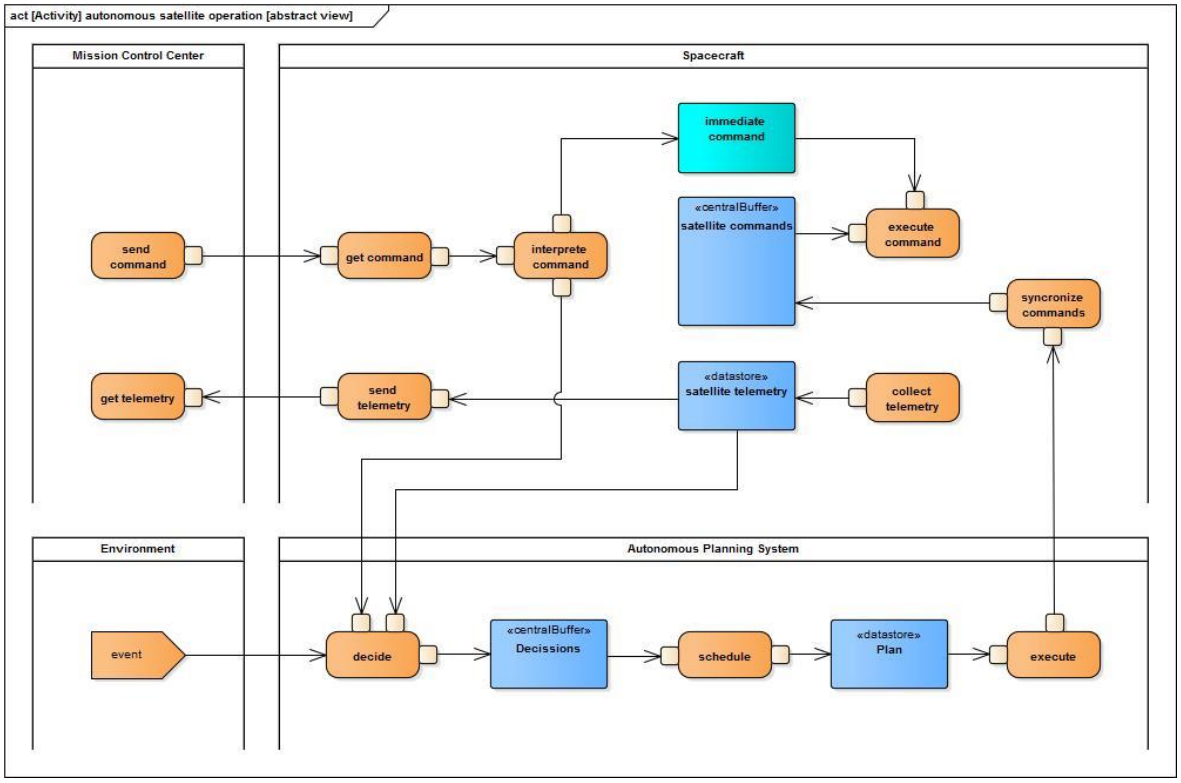


Abbildung 1-7: autonomer Satellitenbetrieb (Vision)

Die Vision zeigt den grundsätzlichen Ansatz, der bei der Lösung des autonomen Planungssystems verfolgt wird. Dabei kommen verschiedene Techniken zum Einsatz. Der nächste Abschnitt soll daher näher erläutern, in welchem Kontext die Dissertation anzusiedeln ist.

1.5 Kontext

In diesem Abschnitt soll die Dissertation in den Kontext der wissenschaftlichen Systematik eingeordnet werden. Dazu müssen auf abstrakter Ebene die Funktionalitäten der autonomen Planung bzw. die Teilprobleme des autonomen Planungsproblems (siehe Abbildung 1-8) betrachtet und anschließend hinsichtlich ihrer Wissenschaftsdisziplin klassifiziert werden:

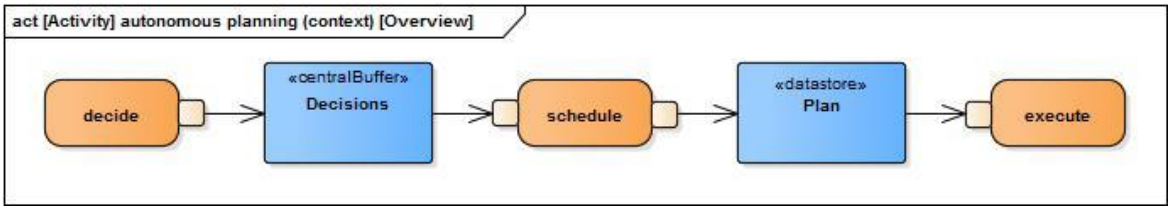


Abbildung 1-8: Funktionalitäten der autonomen Planung

1. Entscheiden:
   Bei der autonomen Planung muss entschieden werden, wie der Satellit auf deterministische und nichtdeterministische Ereignisse reagieren soll. Die Entscheidungen sollen Handlungen des Satelliten bewirken.
2. Planen:
   Für die im vorherigen Schritt beschlossenen Handlungen müssen Zeitpunkte für deren Ausführung festgelegt werden. Dabei müssen auch die zur Verfügung stehenden Ressourcen und Beschränkungen des Satelliten berücksichtigt werden.
3. Ausführen:
   Der zuvor berechnete Plan muss auch umgesetzt werden. Dazu ist es nötig, aus ihm die dafür notwendigen Kommandos abzuleiten. Ein Synchronisierungsmechanismus gleicht diese mit der Kommandoliste der Bordsteuerung ab.

Zum einen ist das autonome Planungsproblem ein auf die Satellitentechnik bezogenes Problem (siehe Referenzszenarien in Abschnitt 7.1) und damit der angewandten Informatik zuzuordnen. Zum anderen ist es auf Grund seiner Konkretheit mit den Mitteln der praktischen Informatik zu behandeln. Wegen der oben beschriebenen Teilprobleme sind dabei folgende Bereiche von besonderem Interesse:

- „Operation Research“
  Dieser Bereich beschäftigt sich mit der Entscheidungsunterstützung zur Lösung von Problemen, die in der Regel kombinatorischer Natur sind. Zu den bekanntesten Problembeispielen [13, p. 1059] gehören das Problem des Handlungsreisenden und das Rucksackproblem. Für das autonome Planungsproblem sind insbesondere die Vorschläge zur Ablaufplanung (Scheduling) von Interesse.
- Künstliche Intelligenz und Software-Agenten
  Dieser Bereich beschäftigt sich mit der Schaffung von Systemen, die in die Lage versetzt werden sollen, in Anlehnung an das menschliche Denken, Probleme zu identifizieren, und einen Lösungsweg eigenmächtig auszuarbeiten und anzuwenden. Diese Vorgehensweise unterscheidet sich von klassischen Informationssystemen, deren Verhalten so entworfen wird, dass sie einen programmierten und vordefinierten Lösungsweg einschlagen. Das Teilgebiet der künstlichen Intelligenz, das sich mit dem Entwurf von Software-Agenten beschäftigt, bietet Ansätze zur Behandlung von Problemen, die einige Parallelen zum autonomen Planungsproblem aufweisen.

Zusammengefasst betrachtet sind Teilaspekte der Fragestellung zum „Autonomen Planungsproblem“ bereits in Bereichen der Informatik aufgetreten. Dementsprechend liegen bereits Erkenntnisse vor. Diese werden im Abschnitt 4 zusammen mit dem Stand der Technik und der Forschung näher ausgewertet. Vorweggenommen zeigt sich jedoch, dass das „Autonome Planungsproblem“ unter Berücksichtigung der Abgrenzung (siehe Abschnitt 1.6) bisher noch nicht gelöst wurde. Der hierfür

notwendige Forschungsaufwand wird durch den zu erwartenden Nutzen (siehe Relevanz im Abschnitt 1.3) gerechtfertigt.

Das Thema Autonomie, insbesondere im Bereich der Raumfahrt, ist wohlbekannt. Deshalb soll die vorliegende Dissertation im nächsten Abschnitt fachlich klar abgegrenzt werden. Auch soll aufgezeigt werden, welche Ziele sie generell verfolgt.

1.6 Zielsetzung und Abgrenzung

Das hauptsächliche Ziel dieser Dissertation besteht darin, eine Möglichkeit aufzuzeigen, wie das autonome Planungsproblem gelöst werden kann. Dazu soll ein autonomes Planungssystem für Kleinstsatelliten zur Steigerung derer autonomen Fähigkeiten konzipiert und umgesetzt werden.

Der Begriff „Kleinstsatellit“ wird in der Raumfahrttechnik, insbesondere im universitären Umfeld, häufig verwendet. Jedoch liegt hierzu bisher keine einschlägige Definition vor. Daher werden in der nachfolgenden Tabelle 1-1 zwei grundlegende Gruppierungen (Klasse und Typ) zur Orientierung vorgegeben.

Gruppierung 1 (Klasse)	**Gruppierung 2 (Typ)**	**Masse (Kg)**	**el. Leistung (W)[2]**	**Volumen (cm³)**
Femto	Kleinst	< 0,1	< 0,2	ca. 2,5 x 2,5 x 0,5
Pico		0,1 – 1,5	< 1,0	ca. 10 x 10 x 10
Nano		1 – 25	< 20	ca. 30 x 30 x 30
Mikro	Klein	10 – 200	< 150	ca. 100 x 100 x 100
Mini		100 – 500	< 500	ca. 200 x 200 x 200

Tabelle 1-1: Eingruppierung der Satelliten hinsichtlich Masse, Volumen und el. Leistung

Zwar befasst sich die Dissertation im Allgemeinen mit der Thematik „Höhere Autonomie“, jedoch liegt der Fokus auf deren Anwendung in Kleinstsatelliten, insbesondere den Nanosatelliten. Hieraus ergibt sich die Problematik, dass im Gegensatz zu größeren Satelliten die zur Verfügung stehenden Ressourcen (zum Beispiel Rechenkapazität, Energie, Leistung, Speicher, …) deutlich eingeschränkt sind. Aus diesem Grund orientiert sich die vorliegende Arbeit besonders an den zwei Zielen „so viel wie nötig“ und „so einfach wie möglich“, um die Ressourcen zu schonen.

Im Laufe des Betriebs eines autonomen Satelliten kann es erforderlich werden, seine Verhaltensweise grundlegend zu ändern. Die übliche Vorgehensweise bestünde darin, unter anderem die komplette (onboard) Software der Satellitensteuerung zu aktualisieren. Dieser Vorgang ist jedoch mit Risiken verbunden, denen der Satellit nicht unbedingt ausgesetzt werden sollte. Aus diesem Grund ist in der Zielsetzung ebenfalls die Forderung enthalten, das autonome Verhalten des Satelliten ausschließlich durch Kommandierung anzupassen.

Eine weitere Abgrenzung ergibt sich daraus, dass diese Dissertation nur Szenarien (mögliche Satellitenmissionen) für das autonome Planungssystem, bei der die Satelliten im Erdorbit kreisen, berücksichtigt. Andere Orbits, zum Beispiel um den Mond oder den Mars, sind nicht berücksichtigt, aber nicht notwendigerweise von der Anwendung ausgeschlossen. Dieselbe Argumentation trifft auf Sonden, Roboter und interplanetare Missionen im Allgemeinen zu.

Die in dieser Dissertation untersuchten Szenarien und Anwendungen für das autonome Planungssystem bestehen nur aus einem einzelnen Satelliten und einer Bodenstation. Dadurch bedingt sind Schwärme von Satelliten und Netzwerke von Bodenstationen nicht Gegenstand der Untersuchung. Wie im vorherigen Fall werden diese Anwendungen aber nicht grundsätzlich ausgeschlossen.

[2] Der Satellit kann in Sonnenphasen mittels Solarzellen Energie aufnehmen. Die effektive Leistungsaufnahme beträgt ca. 1,4 W pro dm² bei direkter Bestrahlung.

2 Fachbegriffe und benötigtes Grundwissen

Dieser Abschnitt hat zum Ziel, den Leser in die wichtigsten in dieser Dissertation verwendeten Fachbegriffe einzuführen, diese erforderlichenfalls näher zu erläutern, und auf entsprechende Literatur zum Nachschlagen zu verweisen. Fachfremden Lesern soll dadurch der Einstieg in die Thematik erleichtert werden.

2.1 Autonomie

In der Literatur der Natur- und Ingenieurswissenschaften ist der Begriff „Autonomie“ nicht eindeutig definiert. Die meisten Bezugnahmen stimmen jedoch darin überein, dass mit dem Begriff eine technische Vorgehensweise bezeichnet wird, bei der Abläufe vollständig automatisch, und ohne das zusätzliche Einwirken von Menschen, ablaufen. Falls die betroffenen Abläufe in ihrem Fluss verzweigen und an den Zweigstellen Entscheidungen nötig werden, werden auch diese selbständig vom autonomen System, das die Abläufe steuert, getroffen.

Die Autonomie soll noch weiter untergliedert werden: in „niedere“ und „höhere“ Autonomie. Niedere Autonomie wird in der englischen Literatur [14, pp. S.11-13] meistens als „autonomic“ bezeichnet, höhere Autonomie hingegen als „autonomous“. Ein System mit niederer Autonomie ist unter anderem mit einer oder mehreren Eigenschaften ausgestattet, wie dem Selbstschutz, der Selbstoptimierung, der Selbstheilung oder der Selbstkonfiguration [14, p. S. 11]. Die niederen autonomen Funktionen haben einen „reflexartigen“ und „reagierenden“ Charakter. Sie dienen hauptsächlich dazu, das System ohne die Mithilfe des Menschen in Funktionsbereitschaft zu halten. Höhere Autonomie hingegen bezieht sich auf die Fähigkeit eines Systems, seine Handlungen zu steuern. In der Regel ist damit die Erstellung eines Plans verbunden, der die nächsten Handlungen des Systems in der Zukunft beschreibt. In konventionellen Systemen wurde diese Aufgabe vom Menschen übernommen. Daher zielt die höhere Autonomie darauf ab, den Menschen als unmittelbaren Entscheidungsträger abzulösen und sein Entscheidungsverhalten zu emulieren. Die Entscheidungsprozesse eines höheren autonomen Systems richten sich nach vom Menschen vorgegebenen „Regeln“ und/oder „Zielen“. Wegen ihres planenden Charakters werden höhere autonome Systeme in der Dissertation auch als autonome Planungssysteme bezeichnet.

2.2 Raumfahrttechnik

Um einen Satelliten mit einem autonomen Planungssystem zu erweitern, ist es zwingend erforderlich, die Hauptakteure der Raumfahrttechnik und ihre Funktionsweisen zu verstehen. Dieser Unterabschnitt soll die dafür notwendigen Grundlagen liefern.

2.2.1 Raumfahrtmissionen und Raumfahrtsegmente

Ausgehend von einem Top-Down-Ansatz, werden die Begriffe „Raumfahrtmission“ und „Raumfahrtsegmente“ inklusive ihrer zugehörigen Teilsysteme näher erläutert. Dieses Grundwissen wurde aus der Literatur [15, pp. S.54-62] und [16, pp. S.4-9] entnommen.

Eine Raumfahrtmission ist ein sehr komplexes Unterfangen mit dem Ziel, eine Aufgabenstellung mit dem Einsatz von Raumfahrtzeugen zu lösen. Bekannte Raumfahrtmissionen sind:

- Raumstationen wie die ISS und die MIR

- Raumsonden, z.B. Cassini-Huygens oder Mars Express
- Satelliten, z.B. Sputnik 1 (1957) oder Envisat (2002), BIRD (2001)

Die Aufgabenstellungen von Raumfahrtmissionen variieren in ihrer Art und dienen damit verschiedenen Zwecken, wie zum Beispiel der Beobachtung, der Telekommunikation, der Wissenschaft und der Technologieerprobung. Ihre Handhabung erstreckt sich über drei Teilbereiche, auch Raumfahrtsegmente genannt (siehe Abbildung 2-1).

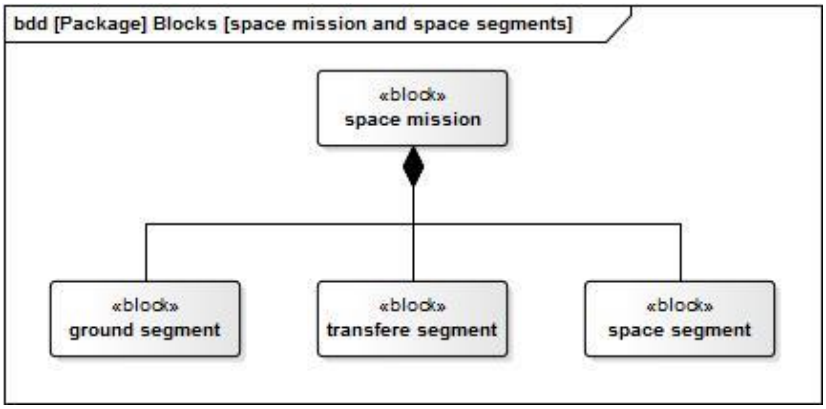


Abbildung 2-1: Raumfahrtmission und Raumfahrtsegmente

Das Raumsegment besteht aus einer Plattform (bei Satelliten auch Satelliten-Bus genannt) und seiner daran montierten Nutzlast. Bei dieser handelt es sich gewöhnlicherweise um Instrumente, die die eigentliche Aufgabenstellung der Raumfahrtmission bewältigen. Die Plattform hat lediglich die Aufgabe, die dafür notwendigen Ressourcen (z.B. elektrische Leistung, mechanische Befestigungsvorrichtungen, thermische Stabilität) für den sicheren Betrieb der Nutzlast zur Verfügung zu stellen. Dem Raumsegment wird, falls es sich tatsächlich um einen Satelliten handelt, ein Orbit zugeordnet, der seine Bahn um einen Zentralkörper festlegt.

Das Raumsegment wird durch das Bodensegment überwacht und kontrolliert. Dieses besteht meistens aus einem Netzwerk aus einer oder mehreren Bodenstationen für den direkten Funkkontakt zum Raumsegment, und einem Kontrollzentrum (evtl. auch einem zweitem als Backup). Das Bodensegment muss unter anderem folgende Aufgaben [14, p. S.26] bewerkstelligen. Die meisten dienen der Unterstützung des Raumfahrtsegments. Dazu gehören unter anderem:

- Planung und Ablaufsteuerung des Satelliten
- Kommandierung des Satelliten
- Erstellung von Zeitplänen für die Nutzlast
- Unterstützung von Onboard-Funktionen des Satelliten (z.B. Kalibrierung der Instrumente)
- Download von Daten und Archivierung (Satelliten-Bus und Nutzlast)
- Performance- und Leistungsüberwachung
- Fehlerdiagnose und Fehlerkorrektur
- Analyse der Daten (Satelliten-Bus)
- Verteilung / Vertrieb der Daten (Nutzlast)

Es ist ersichtlich, dass die Aufgaben des Raumsegments und des Bodensegments eng miteinander verzahnt sind. Daher ist es absehbar, dass ein autonomes Planungssystem sehr stark in die Funktionalitäten beider Segmente eingreifen muss.

Das Transfersegment dient dem Transport des Raumsegments von der Erdoberfläche aus bis in den gewünschten Zielorbit. Typischerweise erfolgt dies durch den Einsatz von Trägerraketen. Da das Raumsegment beim Aufstieg in der Trägerrakete gewöhnlich deaktiviert ist, wird in der Dissertation auf dieses Thema nicht weiter eingegangen.

2.2.2 Subsysteme des Satelliten

Die Plattform des Raumsegments, auch Satelliten-Bus genannt, besteht aus mehreren eng miteinander verzahnten Subsystemen (siehe Abbildung 2-2). Wie bereits erwähnt wurde, liegt ihre Aufgabe darin, der Nutzlast (engl. Payload), die z.B. aus mehreren wissenschaftlichen Instrumenten bestehen kann, ausreichend Ressourcen zur Verfügung zu stellen und ihr die notwendigen Schnittstellen anzubieten.

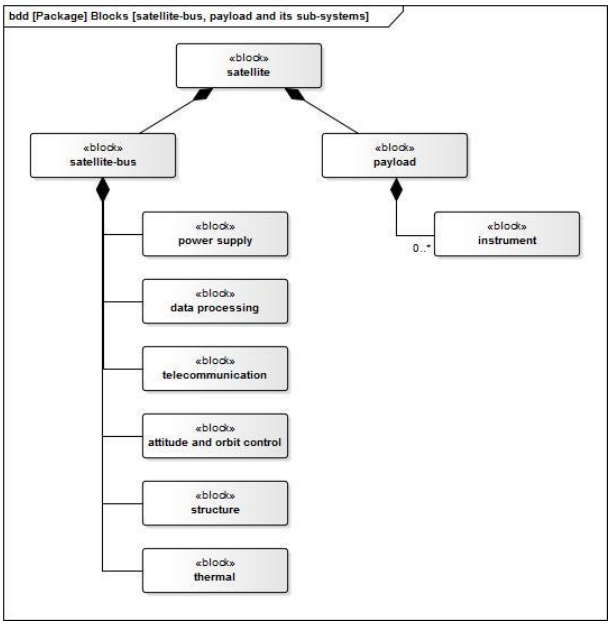


Abbildung 2-2: Sub-Systeme des Satelliten-Bus

Insgesamt gibt es sechs verschiedene Arten von Subsystemen:

Das Subsystem Energieversorgung ist für die Erzeugung und Verteilung elektrischer Energie verantwortlich. Es besteht aus Quellen (i.d.R. Solarzellen), Speichern (z.B. Batterien), Schaltern und Sicherungen. Damit verbunden ist üblicherweise die Regelung elektrischer Ströme und Spannungen. Die Verbraucher des Subsystems für die Energieerzeugung sind die anderen Subsysteme des Satellitenbusses, wie zum Beispiel das Kommunikations-Subsystem oder das Datenverarbeitungs-Subsystem. Das letztere ist für die allgemeine Datenerfassung, -prozessierung, -formatierung und -verteilung im Satelliten verantwortlich, und in der Regel über mehrere Hardwareinstanzen verteilt. Es wird im Abschnitt 2.2.4 gesondert betrachtet, da es von den Eingriffen eines autonomen Planungssystems besonders betroffen wird. Um zwischen einem Satelliten und einer Bodenstation Daten zu übertragen, ist eine Funkverbindung notwendig. Auf dem Satelliten wird diese Aufgabe durch das Kommunikations-Subsystem mit seinen Antennen, Receivern, Transmittern, En- und Decodern übernommen. Befehle an den Satelliten, die über die Funkstrecke gesendet werden, werden als Telekommandos bezeichnet. Daten hingegen, die der Satellit zur Bodenstation sendet, werden unter dem Begriff Telemetrie zusammengefasst. Abschnitt 2.2.5 beschäftigt sich näher mit dieser Thematik und verweist auf die in der Satellitentechnik etablierten Standards für die Datenübertragung per Funk. Ein weiteres Subsystem ist das Bahn- und Lageregelungssystem für die Ausrichtung und die Positionierung des Satelliten im Weltall. Intern arbeitet das Bahn- und Lageregelungssystem nach dem Prinzip einer Regelschleife. Zur Erfassung der Lage werden die Daten diverser Sensoren (z.B. Sternkameras oder Sonnensensoren) ausgelesen und aufgearbeitet. Bei einer Abweichung der gewünschten Lage werden bestimmte Aktuatoren (z.B. Rotationsräder oder Düsen) angesteuert, um den Satelliten in die gewünschte Lage zu bewegen. Die Zielvorgaben für die Lageregelung werden in der Regel durch das Datenverarbeitungs-Modul vorgegeben. Das Struktur-Subsystem stellt das eigentliche mechanische Gerüst samt seinen mechanischen Vorrichtungen zur Befestigung aller Komponenten des Satelliten dar. Es muss mechanische Stabilität und Integrität, auch bei Belastungen, gewährleisten. Das Thermal-Subsystem hat die Aufgabe, sicherzustellen, dass alle Komponenten des Satelliten im zulässigen Temperaturbereich betrieben werden. Dazu müssen evtl. Vorrichtungen, mit denen sich der Thermalhaushalt beeinflussen lässt, wie Heizer oder Strahler, am Satelliten angebracht werden. Das Nutzlast-Subsystem steht stellvertretend für alle Komponenten, Instrumenten und Vorrichtungen, die die eigentliche Aufgabe der Satellitenmission bewerkstelligen.

Es sei noch anzumerken, dass die einzelnen Subsysteme des Satelliten Daten untereinander austauschen. Insbesondere das System für die Datenverarbeitung sammelt Informationen und verteilt sie an die anderen Subsysteme. Auf diese Weise werden Informationen (z.B. über Zustände) erfasst und zwischengespeichert. Diese Informationen sind eine wichtige Ausgangsbasis für ein autonomes

Planungssystem. Bevor jedoch näher auf autonome Planungssysteme eingegangen werden kann, muss zuerst die eigentliche Datenverarbeitung im Satelliten und am Boden näher betrachtet werden. Dies erfolgt im nächsten Abschnitt.

2.2.3 Datenverarbeitung am Boden (Missionsbetrieb)

Der Begriff Missionsbetrieb [15] wird in der Raumfahrttechnik sehr umfassend verwendet und bezieht sich auf alle Prozesse und Einrichtungen, die in den Betrieb des Satelliten involviert sind. Im Abschnitt 1.4 (Vision) wurde ersichtlich, dass das Verschicken von Telekommandos, der Empfang und die Auswertung der Telemetrie dabei von elementarer Bedeutung sind. Zur Realisierung dieser zwei Aufgabenbereiche wird eine Infrastruktur mit speziell abgestimmten Prozessen benötigt. Die nachfolgende Abbildung 2-3 zeigt hierzu einen schematischen Überblick. Je nach Art der Mission, können die stattfindenden Abläufe und die involvierten Einrichtungen von dieser Darstellung abweichen.

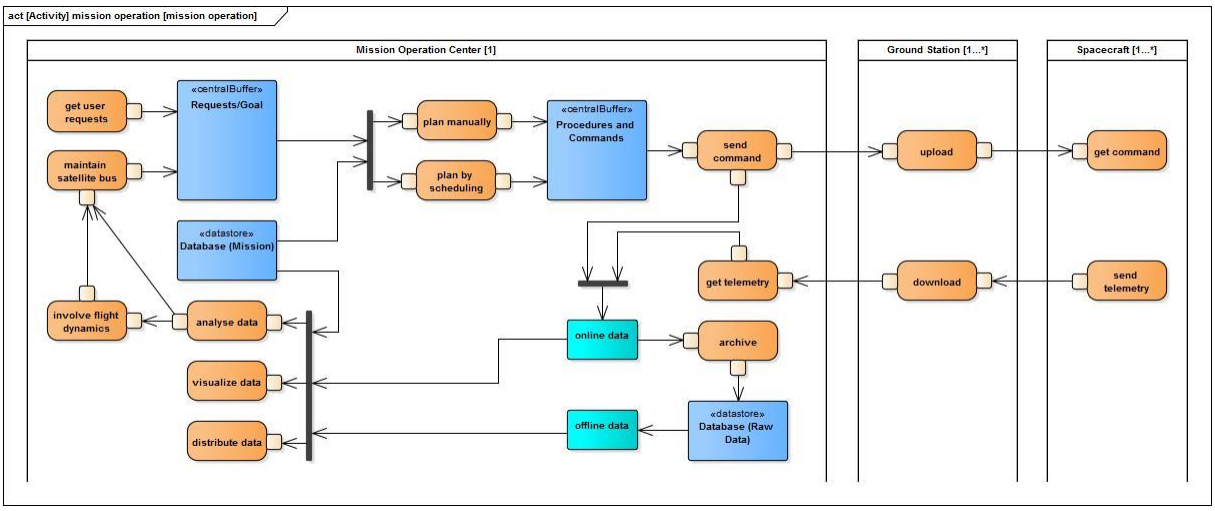


Abbildung 2-3: Aufgaben und Abläufe des Missionsbetriebes

Das Missionskontrollzentrum ist die zentrale Steuereinrichtung für den Satellitenbetrieb. Es nutzt ein Netzwerk von Bodenstationen, um mit dem Satelliten zu kommunizieren, d.h. Telekommandos und Telemetrie zu übertragen. Bei der Erzeugung der Telekommandos greifen mehrere Prozesse ineinander. Zuerst muss von den Nutzern des Satelliten, z.B. Wissenschaftler oder Geschäftskunden, erfasst werden, welche Aufgaben, z.B. wissenschaftliche Messungen oder Erdaufnahmen, der Satellit erledigen soll. Diese Anforderungen werden in Form von Aufträgen gesammelt. Je nach Komplexität der Satellitenmission kann es sein, dass die Erfassung und Verwaltung der Nutzeraufträge auf ein Nutzerkontrollzentrum ausgelagert wird. Dieses trägt dann auch die Verantwortung für die Nutzlast des Satelliten. Die Verantwortung für den Satelliten-Bus hingegen bleibt in der Regel beim Missionskontrollzentrum. Dieses muss sicherstellen, dass der Satellit sich so lange wie möglich im operativen Zustand befindet. Notfalls müssen ferngesteuerte Wartungsarbeiten durchgeführt werden, zum Beispiel in Form von Korrekturmanövern, um Abweichungen vom Zielorbit auszugleichen. In der Regel beschäftigt sich eine Abteilung des Kontrollzentrums ausschließlich mit diesen Thematiken zur Flugdynamik. Direkte vom Kontrollzentrum stammende Anweisungen werden ebenfalls als Aufträge behandelt und mit denen der Nutzer in einer Liste zentral verwaltet. Als nächstes müssen die eigentlichen Kommandosequenzen für den Satelliten generiert werden. Dazu müssen die zuvor zentral gesammelten Aufträge verarbeitet werden. Die dabei auftretende Problemstellung entspricht der eines Planungsproblems, da es hierbei gilt, einen Plan in Form einer Kommandosequenz unter Einhaltung äußerer Beschränkungen, wie sie zum Beispiel dem Satellitensystem auferlegt sind, zu berechnen. Das für die Erstellung notwendige Wissen wird einer Missionsdatenbank entnommen. In dieser sind Informationen enthalten, die beschreiben, welche Kommandosequenzen, in diesem Fall

auch Kommandoprozeduren genannt, der Satellit überhaupt ausführen darf, aus welchen einzelnen Kommandos diese bestehen und welche Ressourcen und Beschränkungen es bei dem Vorgang zu berücksichtigen gilt. Als Ergebnis wird ein Plan in Form einer Kommandosequenz geliefert. Bei komplexeren Missionen wird beim Planungsproblem auf die Hilfe eines Ablaufplaners (Scheduler) zurückgegriffen. Dennoch ist es immer möglich, die Kommandosequenz zusätzlich auch manuell durch den Operator verändern zu lassen. Bei kleineren Satellitenmissionen wird fast ausschließlich diese Methode angewandt. In der Regel werden die Kommandosequenzen zwei Wochen vor ihrer Ausführung geplant und wenige Tage vor ihrer Ausführung zum Satelliten hochgeladen. Der Upload erfolgt über eine der zur Verfügung stehenden Bodenstationen. An Bord werden die Kommandos letztendlich interpretiert, überprüft und schließlich zum vorgegebenen Zeitpunkt ausgeführt. In umgekehrter Richtung produziert und sendet der Satellit Telemetrie in Form von Housekeeping und Nutzlastdaten, die von der Bodenstation empfangen und an das Kontrollzentrum geschickt werden. Dort werden die empfangenen Daten zusammen mit den gesendeten Kommandos prozessiert. Dabei werden je nach Mission ausgewählte Daten zum einem direkt im Anschluss in Echtzeit (Online-Daten) verarbeitet, zum anderen aber auch gleichzeitig archiviert (Offline-Daten). Die Archivierung ermöglicht es, intensivere nachträgliche Datenverarbeitungsprozesse zeitlich von der Kommandierung und der Telemetrierung zu entkoppeln. Unabhängig davon, ob es sich um eine Online- oder Offlinedatenverarbeitung handelt, sind folgende Aufgaben für ein Missionskontrollzentrum typisch:

- Datenanalyse
  Die Telemetriedaten, vor allem die Zustandsdaten (Housekeeping) des Satelliten und seiner Komponenten, werden überwacht und analysiert. Damit lässt sich feststellen, ob sich alle Systeme im zulässigen Bereich befinden. Mit Hilfe zeitlich zurück liegender Daten lässt sich eine Trendanalyse durchführen. Damit können auch mögliche Betriebsprobleme im Satelliten identifiziert werden, die sich zwar anbahnen, aber noch nicht eingetreten sind. Sollten auf Grund der Analyse sofortige oder künftige Maßnahmen für den Satelliten erforderlich sein, so können diese in den Planungsprozess für die Kommandierung einfließen.
- Datenvisualisierung
  Für den Operator ist es zur besseren Handhabung sehr hilfreich, wenn die empfangenen und gesendeten Daten (Kommandos und Telemetrie) in einer intuitiven und leicht verständlichen Form dargestellt werden. Hierzu wurden verschiedene Anzeigetechniken entwickelt. Notfalls müssen auch Rohdaten in eine passende Darstellung transformiert werden.
- Datenverteilung
  Das Missionskontrollzentrum muss über eine passende Infrastruktur verfügen, um den eigentlichen Nutzern des Satelliten einen Zugang zu den angeforderten Daten (zum Beispiel Bilddaten) zu ermöglichen.

2.2.4 Datenverarbeitung im Satelliten

Die wichtigste Steuerkomponente im Satelliten ist der Bordcomputer (engl. Onboard Computer, OBC). Dieser ist meistens direkt mit den einzelnen Subsystemen bzw. Komponenten des Satelliten verbunden und für dessen zentrale Steuerung verantwortlich. Wenn der Satellit mit autonomen Fähigkeiten erweitert werden soll, ist eine direkte Interaktion mit dem OBC unerlässlich, da dies die einzige Möglichkeit darstellt, das Verhalten des Satelliten zu verändern. Aus diesem Grund sollen in diesem Unterabschnitt die generelle Funktionsweise und die Kommandierungsabläufe im Bordcomputer näher erläutert werden.

2.2.4.1 Bordcomputer

Obwohl sich die Satellitenmissionen untereinander zum Teil sehr unterscheiden, weisen sie einige Gemeinsamkeiten in der Software-Architektur ihrer Bordcomputer auf. Meistens sind diese nach einem Schema, wie in Abbildung 2-4 dargestellt, konzipiert.

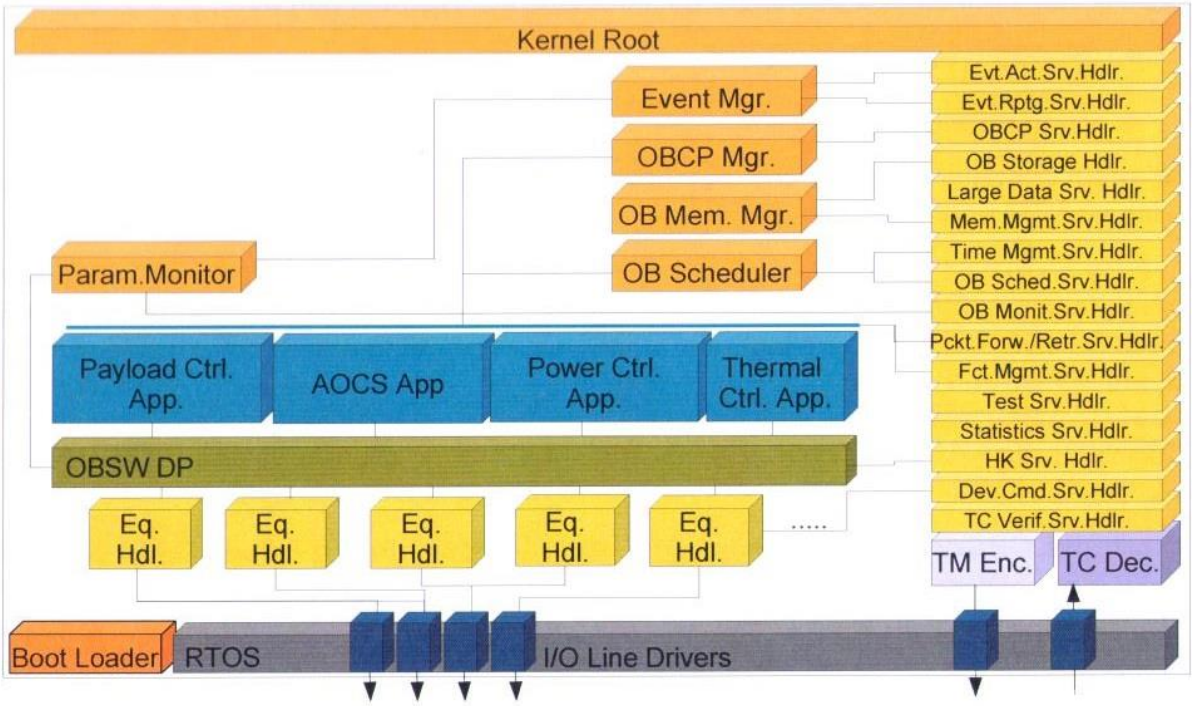


Abbildung 2-4: Software-Komponenten eines Bordcomputers [10, p. S.111]

Die Software eines Bordcomputers ist in der Regel vierschichtig aufgebaut:

1. Hardware-Abstrahierungsschicht
2. Datenschicht
3. Anwendungsschicht
4. Serviceschicht

Auf der untersten Ebene, der HW-Abstrahierungsschicht, befinden sich die hardwarenahen Komponenten, wie zum Beispiel der Bootloader, das Echtzeitbetriebssystem (Realtime Operating System, RTOS) und diverse Hardwaretreiber für externe Komponenten. Mit dem Bootloader wird das Programm beim Hochfahren der Hardware in den Hauptspeicher geladen und initialisiert. Nach Abschluss geht der Kontrollfluss an das Echtzeitbetriebssystem über. Je nach Komplexität ist es Aufgabe des RTOS, die Ressourcen (z.B. Speicher) zu managen und mit Hilfe seiner internen Ablaufsteuerung die diversen Handler, Threads, Prozesse und Applikationen aufzurufen. Auf der Hardware-Abstrahierungsschicht baut die Datenschicht auf. Sie besteht aus verschiedenen Handlern für die externen Komponenten und einem globalen Speicher (Data Pool, DP). Die Handler bekommen als Input die von den Treibern gelieferten Rohwerte, prozessieren (z.B. kalibrieren) diese, und legen die Ergebnisse im globalen Speicher ab. Die Softwarekomponenten der nächsten Schicht, auch Anwendungen genannt, greifen auf die Daten im globalen Speicher zu, führen damit höhere Berechnungen durch, und legen ihre Ergebnisse wieder im globalen Speicher ab. Sie besitzen eigene Instanzen an Software-Konstrukten, wie zum Beispiel Threads, Prozesse, Stacks, Heaps, usw. Einige Anwendungen arbeiten meistens im Kontext eines Subsystems des Satelliten (z.B. Lageregelung). Andere hingegen sind eher zentraler Natur und arbeiten auf breiterer Basis. Besonders hervorzuheben sind folgende Anwendungen:

- Manager für die Ereignisse
- Manager für die Kontrollprozeduren (Onboard Control Procedure, OBCP)
- Manager für die Speicherverwaltung
- Manager für die Ablaufsteuerung (Scheduler) der Kommandos
- Manager für die Überwachung des globalen Speichers

Ereignisse sind in diesem Sinne besondere Vorkommnisse, die eine gesonderte Benachrichtigung der Bodenstation erfordern. Der verantwortliche Manager unternimmt die dafür notwendigen Schritte. Solch ein Ereignis kann zum Beispiel ausgelöst werden, wenn ein bestimmter Wert im globalen Speicher einen Schwellwert über- oder unterschreitet. Derartige Regelverletzungen werden vom Manager für die Überwachung des globalen Speichers festgestellt und dem Ereignismanager zur Behandlung mitgeteilt. Die Manager für die Kontrollprozeduren und die Kommandoablaufsteuerung werden im nächsten Abschnitt (2.2.4.2 Kommandierung) näher beschrieben, da sie für die Integration eines autonomen Planungssystems von essentieller Bedeutung sind. Die letzte zu erläuternde Schicht ist die Serviceschicht. Sie dient als Schnittstelle zum Kommunikations-Subsystem des Satelliten. Ihre wichtigste Aufgabe besteht darin, die übermittelten Kommandos nach der Dekodierung zu interpretieren und ggf. mit den Softwarekomponenten der anderen Schichten zu interagieren, um die Interpretation abzuarbeiten. Ebenso muss die Serviceschicht in die andere Richtung ausgehende Botschaften zum Telemetrie-Encoder weiterreichen. Für die Serviceschicht hat sich in der Raumfahrt, insbesondere für die ESA, der „Packet Utilization Standard“ (PUS) als Standard etabliert [17]. Dieser umfasst folgende Services [10, p. S. 102]:

- Verifikation der Telekommandos
- Verteilung der Kommandos
- Housekeeping & Berichterstattung der Diagnose
- Statistiken der Parameter
- Berichterstattung über Ereignisse
- Speicherverwaltung
- Funktionsverwaltung
- Zeitverwaltung
- Ablaufsteuerung der Operationen
- Überwachung
- Transfer von großen Datenmengen
- Weiterleitung von Datenpaketen
- Speicherung und Wiederherstellung
- Test
- Operationen und Prozeduren
- Ereignisbehandlung

Aufgrund seines Umfangs wird dieser Standard jedoch nur in größeren Satellitenmissionen vollständig umgesetzt. Kleinere Satellitenmissionen implementieren ihre eigenen angepassten Serviceschichten oder verzichten sogar ganz auf diese, indem Kommandos und Telemetrie direkt von einer dedizierten Anwendungskomponente verwaltet und interpretiert werden.

In den bisherigen Erläuterungen wurden die Softwarekomponenten und Services, die die Kommandierung des Satelliten direkt betreffen, auf Grund ihrer Bedeutung bewusst zurückgestellt. Sie sollen gesondert im nächsten Abschnitt beschrieben werden.

2.2.4.2 Kommandierung und Steuerung des Satelliten

Der Zweck der Kommandierung besteht in der Ausführung bestimmter Handlungen im Satelliten. Diese dienen überwiegend der Steuerung. Dabei muss berücksichtigt werden, dass sich der Satellit beim operationellen Betrieb im Orbit befindet und Anweisungen an ihn daher über Funk übertragen

werden müssen. Die nachfolgende Abbildung 2-5 zeigt die grundsätzlichen Arbeitsschritte bei der Kommandierung auf.

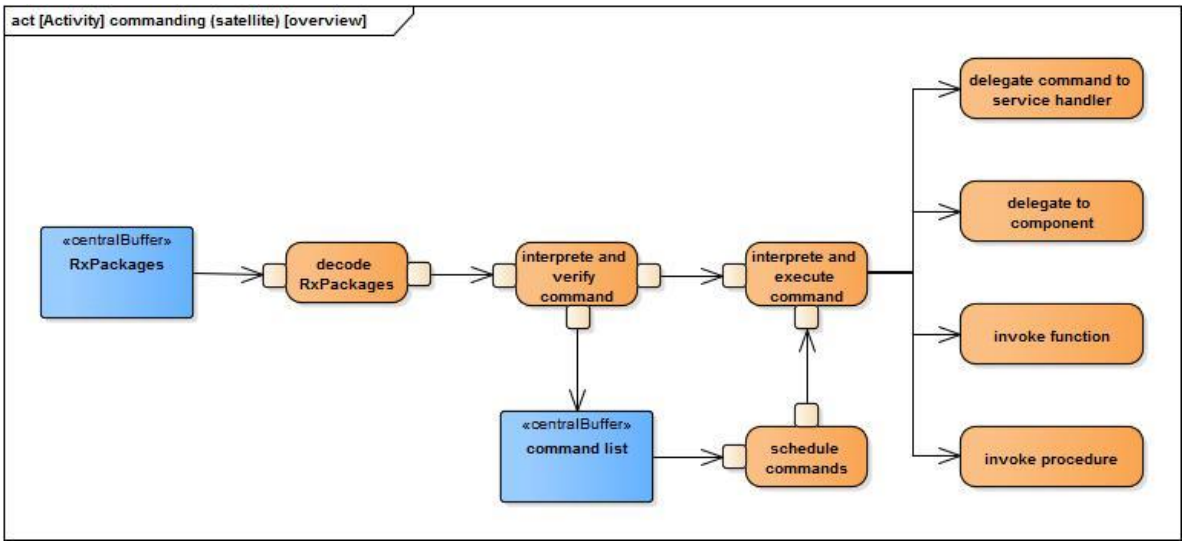


Abbildung 2-5: Grundlegende Arbeitsschritte bei der Kommandierung

Bei der Kommandierung ist das einzelne Kommando von zentraler Bedeutung. Hierbei handelt es sich um ein Datenkonstrukt, in dem Anweisungen für den Satelliten kodiert sind. Die Bodenstation verschickt Kommandos an den Satelliten, indem sie diese gemäß einem Übertragungsprotokoll (siehe Abschnitt 2.2.5) in Datenpakete verpackt und über die Funkstrecke zum Satelliten verschickt. Dort werden die Pakete nach einer Zwischenspeicherung wieder entpackt (Übertragungsprotokoll). Die dabei wiederhergestellten Kommandos werden in einem nächsten Schritt auf Korrektheit hin überprüft und interpretiert. Bei der Interpretation wird festgestellt, ob das Kommando sofort (engl. immediate command) ausgeführt werden soll oder erst zu einem späteren bestimmten Zeitpunkt (engl. time-tagged command). Im letzteren Fall werden die betroffenen Kommandos in einer speziellen Kommandoliste zwischengespeichert. Ein besonderer Ablaufplaner (Scheduler) greift auf diese Kommandoliste zu und veranlasst die Ausführung der entsprechenden Kommandos, falls deren Bedingungen (meistens der zuvor erwähnte Zeitpunkt) erfüllt werden. Je nach Kodierung des auszuführenden Kommandos wird in der Regel eine der vier im Folgenden beschriebenen Handlungen ausgeführt:

1. Das Kommando wird an einen Service-Handler zur Ausführung weitergereicht. Dieser Fall setzt die Implementierung einer Service-Schicht (siehe Abschnitt 2.2.4.1) voraus.
2. Das Kommando wird direkt an eine Hardware- oder Softwarekomponente des Satelliten weitergereicht. Diese Möglichkeit wird oft genutzt, wenn die Komponenten in der Lage sind, die Kommandos selbst zu interpretieren. Die Sternkamera STELLA (siehe Abschnitt 1.2.3) ist ein passendes Beispiel dafür.
3. Das Kommando löst eine vorkompilierte (onboard) Funktion im Bordcomputer aus. Diese Funktionen sind starr in ihrem Ablauf und bieten wenig Flexibilität. Beispiele hierfür sind das Ein- oder Ausschalten bestimmter Komponenten im Satelliten.
4. Das Kommando löst eine (onboard) Prozedur [10, pp. S.126-128] im Bordcomputer aus. Diese Prozeduren sind im Vergleich zu vorkompilierten Funktionen wesentlich flexibler. Je nach Umsetzung bieten sie verschiedene Möglichkeiten, angefangen bei der Parameterübergabe bis hin zur gezielten Steuerung des Kontrollflusses der Prozedur. Meistens sind diese Prozeduren in einer speziellen Skriptsprache geschrieben, die zur Ausführung einen Interpreter benötigen. Die Umsetzung von Prozeduren ist sehr komplex und in der Regel stark eingeschränkt. Beispiele sind nur in größeren Satellitenmission vorzufinden.

Funktionen und Prozeduren bieten die einzige Möglichkeit im Kontext der gesamten Bordcomputer-software (d.h. Handler, Anwendungen, globaler Speicher, Threads, Prozesse, …), Handlungen

durchzuführen. Da diese aber im Fehlerfall gravierende, wenn nicht sogar fatale, Auswirkungen auf den Satelliten haben können, werden sie während der Entwicklung gründlich getestet.

Eine weitere Begrifflichkeit, die für die Steuerung des Satelliten von essentieller Bedeutung ist, ist die der Flugprozeduren. Hierbei handelt es sich um am Boden erstellte und verwaltete Sequenzen von Kommandos, die als eine Einheit betrachtet werden, und den Zweck haben, bestimmte zusammenhängende Aktionen im Satelliten auszulösen. Der Bordcomputer kennt das Konzept der Flugprozeduren nicht und behandelt daher jedes Kommando in einer solchen Prozedur getrennt. Sie werden daher nur am Boden verwaltet und verwendet. Sie dienen zur Spezifikation zusammenhängender Aufgaben im Satelliten und müssen als atomare Einheit betrachtet werden. Zur Steuerung des Satelliten dürfen Operatoren nur getestete und verifizierte Flugprozeduren verwenden. Um zum Beispiel die Sternkamera STELLA (siehe Abschnitt 1.2.3) vom Boden aus im Satelliten zu aktivieren, müsste folgende Flugprozedur (bestehend aus mehreren Kommandos) zum Satelliten gesendet werden:

1. Schalte die Energieversorgung (Spannung und Strom) für STELLA ein
2. Übertrage die notwendigen Parameter an STELLA
3. Aktiviere den Algorithmus von STELLA für die Lageerkennung

Bei der Kommandierung werden Datenpakete von der Bodenstation aus zum Satelliten geschickt. In bestimmten Situationen allerdings muss der Satellit Daten auch in die entgegengesetzte Richtung senden. Diese Thematik soll im nächsten Abschnitt näher erläutert werden.

2.2.4.3 Telemetrierung und Überwachung des Satelliten

Es ist zwingend notwendig, dass der Satellit in der Lage ist, Daten zur Bodenstation zu senden. Zum einen möchte das Flugteam am besten jederzeit über den aktuellen Zustand des Satelliten, seiner Sub-Systeme und seiner Komponenten informiert sein. Zum anderen müssen die mit der Nutzlast gewonnenen Daten zur weiteren Auswertung ebenfalls zur Bodenstation gesendet werden. Die zum Boden zu übermittelnden Daten werden allgemein in der Satellitentechnik als Telemetrie bezeichnet. Der Vorgang, der dies bewerkstelligt, wird in der vorliegenden Dissertation als Telemetrierung bezeichnet. Die nachfolgende Abbildung 2-6 zeigt die einzelnen Arbeitsschritte bei diesem Vorgang.

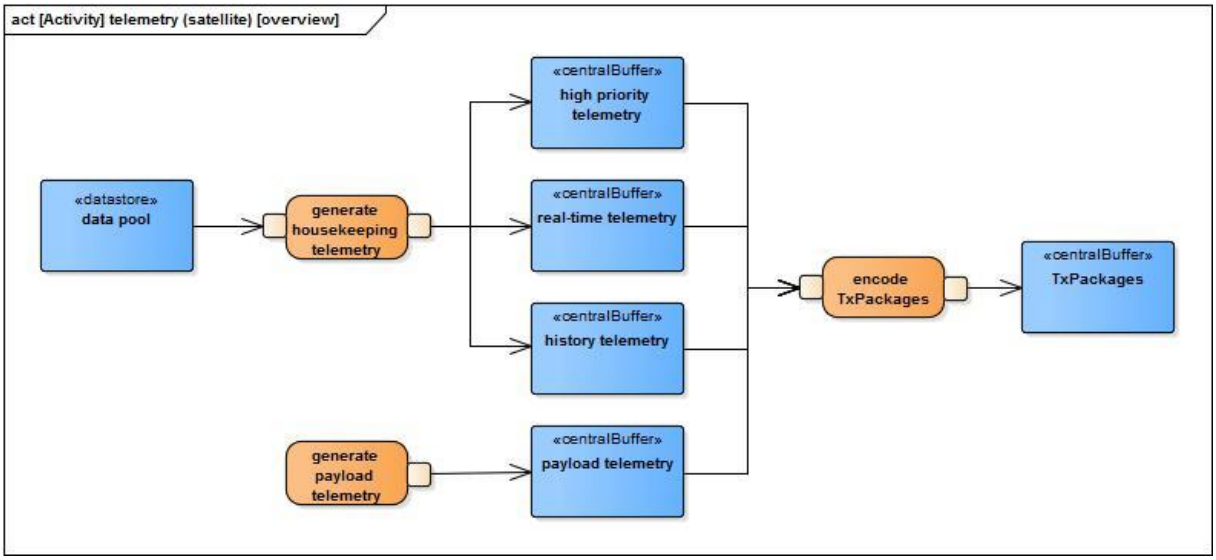


Abbildung 2-6: Grundlegende Arbeitsschritte bei der Erstellung von Telemetrie

Grundsätzlich ist das Team am Boden an vier Arten von Telemetrie interessiert [10, pp. S.113-114]:

1. Echtzeit-Telemetrie
2. History-Telemetrie
3. Dringlichkeits-Telemetrie
4. Nutzlast-Telemetrie

Bei der Echtzeit-Telemetrie handelt es sich um aktuelle Informationen zum Zustand des Satelliten und seiner Komponenten (z.B. freie Speicherkapazitäten, Betriebsmodi diverser Komponenten, usw.) einschließlich der Nutzlast. Bei bestehendem Funkkontakt zwischen dem Satelliten und der Bodenstation wird die Echtzeit-Telemetrie so schnell wie möglich dem Operationsteam zugänglich gemacht. Sie dient hauptsächlich der Überwachung und dem schnellen Eingreifen. Geschehnisse im Satelliten können somit live mitverfolgt werden. Kritische Kommandierungen werden aus diesem Grund immer dann nur durchgeführt, wenn Echtzeit-Telemetrie verfügbar ist. Die History-Telemetrie hingegen wird in der Regel während der gesamten Missionsdauer des Satelliten permanent erstellt und bei Kontakt heruntergeladen. Sie wird jedoch mit geringerer Abtastrate als die Echtzeit-Telemetrie erstellt und dient lediglich der nachträglichen Aufbereitung der Mission. Manche Satelliten senden auch sogenannte Dringlichkeits-Telemetrie. Diese wird verwendet, um auf einen besonderen Zustand im Satelliten hinzuweisen, der in der Regel das Eingreifen eines Operators erfordert. Zuletzt sei noch die Nutzlast-Telemetrie zu erwähnen. Diese enthält die von der Nutzlast gesammelten Daten (zum Beispiel Bilddaten oder Messwerte von Instrumenten).

Im Bordcomputer gibt es diverse Applikationen, Handler und Services, die die verschiedenen Arten von Telemetrie generieren können. Die zu übermittelnden Daten werden vorläufig in besonderen Speicherbereichen zwischengespeichert. Bei der nachfolgenden Datenübertragung zur Bodenstation wird die Telemetrie aus dem Speicher entnommen, gemäß einem Datenübertragungsprotokoll verpackt (siehe Abschnitt 2.2.5) und zur Bodenstation gesendet.

Die Abläufe der Telemetrierung sind für ein autonomes Planungssystem von essentieller Bedeutung, da alle Informationen bzgl. der Zustandserfassung aller Komponenten im Satelliten an einer Stelle zentral vereint vorzufinden sind.

In diesem und dem vorherigen Abschnitt wurde erwähnt, dass die Datenübertragung zwischen Satelliten und der Bodenstation gemäß einem Protokoll stattfindet. Dieses soll im nächsten Abschnitt näher vorgestellt werden.

2.2.5 Datenübertragung zwischen Satelliten und Bodenstation

Für die Kommandierung und Telemetrierung eines Satelliten ist eine sicher funktionierende Funkverbindung zwingende Voraussetzung. Während für die Kommandierung Kommandos von der Bodenstation über die Funkstrecke zum Satelliten gesendet werden, erfolgt der Datenverkehr bei der Telemetrierung in die andere Richtung. Telemetrie wird vom Satelliten aus zur Bodenstation gesendet. Für die diversen Komponenten im Satelliten und in der Bodenstation, die in der Lage sind, Kommandos und Telemetrie zu generieren bzw. zu interpretieren, ist es irrelevant, auf welcher Art und Weise dies geschieht. Es muss lediglich gewährleistet sein, dass die Daten unverfälscht an ihrem Bestimmungsort ankommen. Diese Aufgabe wird im Satelliten vom Kommunikations-Submodul übernommen. Es setzt eine Reihe diverser Übertragungsprotokolle ein, um die Daten für die Übertragung vorzubereiten. Die Protokolle führen zum Beispiel Fehlerkorrekturen oder Datensegmentierungen durch. Physikalisch werden die Datenpakete über die Antennen der Transmitter und der Receiver abgestrahlt bzw. empfangen.

Das „Consultative Committee for Space Data Systems" (CCSDS) befasst sich mit dieser Thematik und schlägt verschiedene Protokolle, die speziell für den Einsatz in Weltraumanwendungen konzipiert

wurden, als Standard vor. Die nachfolgende Abbildung 2-7 zeigt den typischen vom CCSDS vorgeschlagenen Protokollstapel [18], der sich am OSI/ISO-Schichtenmodell orientiert.

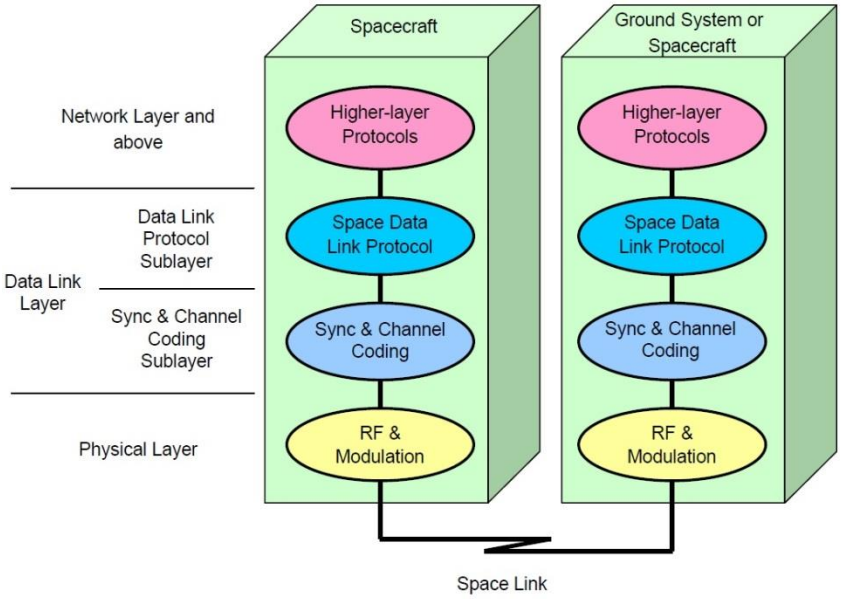


Abbildung 2-7: Datenübertragung nach CCSDS [18, p. S. 2.4]

Im Folgenden werden ausgewählte und wichtige Features der Datenvermittlungsschicht vorgestellt:

- Überprüfung der Datenintegrität mittels Checksummen
- Flag-Informationen bzgl. des Kommandos (z.B. angenommen, in Ausführung, …)
- Routing-Information zur richtigen Weiterleitung der Datenpakete zum Service-Handler
- Segmentierung
- Encodierung
- Sequenzzähler
- Zeitstempel
- Priorisierung (mittels Flags bei Kommandos)
- Priorisierung (mittels virtueller Kanäle bei Telemetrie)
- Erneutes Senden im Fehlerfall

2.3 Ablaufplanung

Ein Teilaspekt des allgemeinen autonomen Planungsproblems ist das sogenannte Ablaufplanungsproblem (scheduling problem). Um dieses beschreiben zu können, müssen folgende Begriffe eingeführt werden:

- Aktivitäten (Activities)
- Ressourcen (Resources)
- Beschränkungen (Constraints)

Im Bereich der Ablaufplanung wird der Begriff „Aktivität“ sehr abstrakt verwendet. Er steht stellvertretend für die Möglichkeit, eine Wirkung von zeitlich begrenzter Dauer auf eine Sache / einen Gegenstand / ein Objekt zu entfalten. Im Folgenden sollen einige konkrete Beispiele für Aktivitäten in diesem Sinne genannt werden:

- Prozeduren einer Steuerung (z.B. Bordcomputer eines Satelliten)

- Funktionen einer Programmiersprache oder eines ausführbaren Programmes
- Aufgaben (Tasks) und Prozesse eines Betriebssystems
- Handlung eines Systems

Je nach Problem, für das eine Ablaufplanung erstellt werden soll, kann es erforderlich sein, sogenannte Ressourcen miteinzubeziehen. Auch dieser Begriff ist abstrakt. Ressourcen weisen eine bestimmte Menge von Sachen / Gegenständen / Objekten auf, die durch die Ausführung der Aktivitäten beeinflusst werden kann. Dabei können Einheiten einer Ressource entweder verbraucht oder produziert werden.

Die Aufgabe eines Ablaufplaners (Scheduler) besteht darin, aus einer vorgegebenen Menge von in Auftrag gegebenen Aktivitäten und Ressourcen einen Plan zu erstellen, der genau beschreibt, zu welchem Zeitpunkt eine Aktivität ausgeführt wird. Für die Berechnung kommen sogenannte Ablaufplanungsalgorithmen (scheduling algorithms) zum Einsatz. Besonderheit dabei ist, dass in der Regel die Aktivitäten und Ressourcen bestimmten Beschränkungen unterworfen sind. Zum Beispiel kann definiert werden, dass Aktivitäten innerhalb eines vorgegebenen Zeitraumes mit einer festgelegten Dauer ausgeführt werden müssen. Die Ressourcen dürfen dabei bestimmte Unter- und Obergrenzen nicht verletzen. Die Algorithmen selbst orientieren sich an zuvor festgelegten Kriterien. Meistens versuchen sie, eine Optimierung des Plans (z.B. die Vermeidung von Ausfallzeiten) vorzunehmen. Selbstverständlich muss der generierte Plan zu den ihm auferlegten Beschränkungen konform sein.

Die nachfolgende Abbildung 2-8 veranschaulicht den Prozess zur Ablaufplanung.

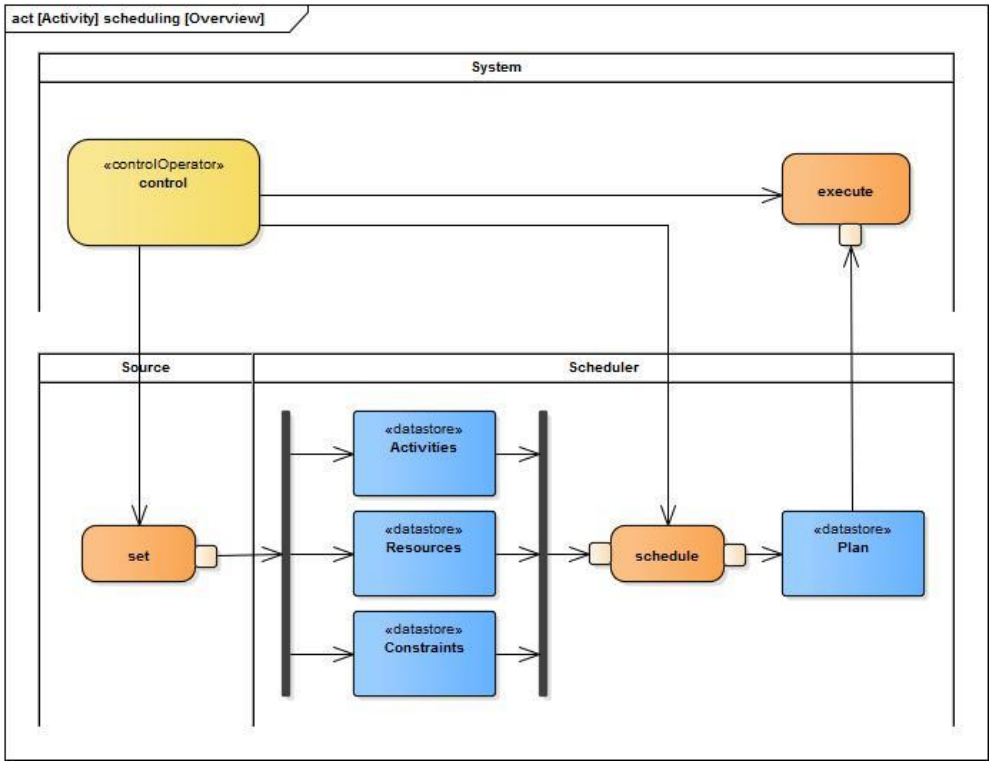


Abbildung 2-8: Prozess zur Ablaufplanung

Die Ablaufplanung ist meistens nur Teil eines Gesamtsystems. Ihre einzige Aufgabe besteht darin, Ablaufpläne zu berechnen. Dazu wird sie von Quellen mit den nötigen Eingangsinformationen (Aktivitäten, Ressourcen und Beschränkungen) versorgt. Der aus der Ablaufplanung resultierende Plan wird im weiteren Verlauf von dem System ausgeführt. Eine Steuerung leitet dabei den Kontroll- und Datenfluss.

Bis jetzt wurden die Teilaspekte des Ablaufplanungsproblems (Aktivitäten, Ressourcen und Beschränkungen) behandelt. Damit für konkrete Probleme passende Ablaufplanungsalgorithmen ausgesucht bzw. entwickelt werden können, müssen die abstrakten Größen in einem mathematischen Modell formalisiert werden. Es ist hilfreich, zuvor eine Klassifikation der Ablaufplanungsprobleme und der zur Lösungsfindung vorgesehenen Algorithmen vorzunehmen. Im Forschungsbereich des „Operation Researchs“ wurden bereits einige Modelle vorgeschlagen. Im Folgenden soll ein Beispiel für eine Formalisierung gezeigt werden, die die Grundlage für die weiteren Arbeiten im Abschnitt 7 (Realisierung) bildet. Das Beispiel orientiert sich an den Vorschlägen in [19], [20], [21], [22].

Ablaufprobleme werden zunächst danach unterschieden, ob sie statisch oder nichtstatisch sind. Statische Ablaufprobleme haben die Eigenschaft, dass die zu planenden Aktivitäten, Ressourcen und Beschränkungen vor und während der Laufzeit feststehen, und sich auch im laufenden Betrieb nicht ändern. Bei dynamischen Ablaufproblemen gilt das nicht. In diesem Fall muss der Algorithmus laufend sich ändernde Eingangsdaten akzeptieren, und den Plan dementsprechend anpassen.

Ein weiteres wichtiges Unterscheidungsmerkmal ist die Präemption. Eine präemptive Ablaufplanung erlaubt die Unterbrechung bereits laufender Aktivitäten und deren Fortführung zu einem späteren Zeitpunkt. Bei nichtpräemptiven Plänen sind Unterbrechungen generell nicht zulässig.

A bezeichnet die Menge aller zu planenden Aktivitäten mit $a_i \in A$ und $i = 1, \dots, |A|$.

Eine Aktivität $a_i \in A$ besitzt folgende Beschränkungen:

- Frühester Startzeitpunkt (release time): $rt(a_i)$
  Der Zeitpunkt, zu dem die Aktivität am frühesten beginnen darf.
- Spätester Endzeitpunkt (deadline time): $dt(a_i)$
  Der Zeitpunkt, zu dem die Aktivität am spätesten abgeschlossen sein muss.
- Prozessdauer (processing time): $pt(a_i)$
  Die Zeitdauer, die die Aktivität für ihre erfolgreiche Ausführung benötigt.

Im Rahmen der Ablaufplanung werden jeder erfolgreich verplanten Aktivität zusätzlich folgende Eigenschaften zugewiesen:

- Tatsächlicher Startzeitpunkt (initial start time): $ist(a_i)$
  Der Zeitpunkt, zu dem die Aktivität zum ersten Mal tatsächlich gestartet wurde.
- Tatsächlicher Endzeitpunkt (final end time): $fet(a_i)$
  Der Zeitpunkt, zu dem die Aktivität vollständig abgeschlossen wurde.

Für den Fall, dass die Ablaufplanung Präemption unterstützt, muss das Modell der Aktivitäten noch um weitere Eigenschaften ergänzt werden:

Der Ausdruck n_{max} definiert mit $n = 1, \dots, n_{max}$ und $n \in \mathbb{N}$ die Anzahl der Unterbrechungen der Aktivität a_i:

- Fortlaufender Startzeitpunkt (continuous start time): $cst(a_i, n)$
  Der Zeitpunkt, zu dem die Aktivität nach n Unterbrechungen wieder ihre Arbeit aufnimmt.
- Fortlaufender Endzeitpunkt (continuous end time): $cet(a_i, n)$
  Der Zeitpunkt, zu dem die Aktivität zur n –ten Unterbrechung angehalten wurde.
- Aktive Gesamtlaufzeit (active time): $at(a_i)$
  Die Summe $\sum_{n=1}^{n_{max}} cet(a_i, n) - cst(a_i, n-1)$ aller aktiven Laufzeiten zwischen $cet(a_i, n)$ und $cst(a_i, n)$ für $n = 1, \dots, n_{max}$

Die nachfolgende Abbildung 2-9 zeigt je ein Beispiel einer nichtpräemptiven und einer präemptiven geplanten Aktivität mit $n_{max} = 2$.

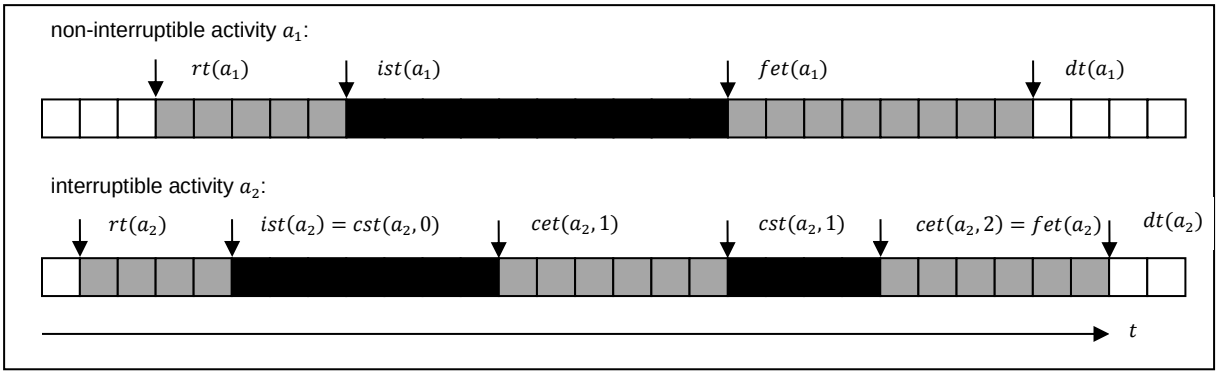


Abbildung 2-9: Beispiele einer nichtpräemptiven und einer präemptiven geplanten Aktivität

R bezeichnet die Menge aller Ressourcen $r_i \in R$ mit $i = 1, \dots, |R|$.

Eine Ressource $r_i \in R$ besitzt folgende Eigenschaften:

- Kapazität (capacity): $cap(r_i)$
  Die maximale Menge an Einheiten einer Ressource r_i, auf die jede beliebige Aktivität $a_j \in A$ zugreifen kann
- Kosten (cost) $cost(r_i, a_j)$
  Eine Funktion, die die Kosten definiert, wenn eine Aktivität $a_j \in A$ auf der Ressource $r_i \in R$ läuft.
- Gewichtung (weight): ($weight(r_i, a_j)$
  Eine Funktion, die den Verbrauch/Produktion der Ressource r_i durch die Aktivität $a_j \in A$ definiert. Die Summe $\sum_{j=1}^{|A|} weight(r_i, a_j)$ aller Gewichtungen zur Ressource r_i darf deren Kapazität $cap(r_i)$ nicht überschreiten.
- Dauer (duration): $dur(r_i, a_j) = pt(a_j)$
  Eine Funktion, die die benötigte Zeit definiert, wenn die Aktivität $a_j \in A$ auf der Ressource r_i ausgeführt wird.

In der Regel versuchen Ablaufplanungsalgorithmen den Plan hinsichtlich bestimmter Kriterien zu optimieren. Zum Beispiel:

- Minimierung der Abarbeitungszeit: $\min(\sum_{j=1}^{|A|}(fet(a_j) - ist(a_j)))$
- Minimierung der Wartezeit: $\min(\sum_{j=1}^{|A|}(fet(a_j) - rt(a_j) - pt(a_j)))$

Die Wahl eines geeigneten Ablaufplanungsalgorithmus hängt von der Art des Ablaufplanungsproblems und der gewünschten Optimierung ab. An dieser Stelle soll auf die entsprechende Fachliteratur [19], [20], [21], [22] verwiesen werden. Die nachfolgende Liste nennt einige bekannte Algorithmen zur Lösung von Ablaufplanungsproblemen:

- As Soon As Possible (ASAP)
- List Scheduling
- Forced Directed Scheduling
- First Come First Served (FCFS)
- Shortest Job First (SJF)

2.4 Modellierungssprache SysML

Den Großteil der in dieser Dissertation enthaltenen Abbildungen stellen Diagramme dar, die konform zur graphischen Modellierungssprache „System Modeling Language“ (SysML) in der Version 1.3 sind. Die SysML ist eine Erweiterung der bekannten Modellierungssprache „Unified Modeling Language“ (UML). Während sich die UML ausschließlich auf reine Softwaretechnik beschränkt, deckt die SysML den gesamten Systementwurf inkl. Hardware und Software ab. Charakteristisch für SysML-Diagramme sind die Rahmen, in deren oberen linken Ecke der Diagrammtyp mit zwei bis drei Buchstaben angegeben ist. In der vorliegenden Dissertation werden folgende Diagrammtypen verwendet:

- Anforderungs-Diagramm (Requirement Diagram - req):
  Dient der Darstellung von Anforderungen. Diese bilden die Ausgangsbasis des Entwurfs von Systemen.
- Anwendungsfall-Diagramm (Use Case Diagram - uc):
  Dient der Darstellung von Anwendungsfällen. Anwendungsfälle entsprechen Funktionalitäten, die von einem System der Umwelt angeboten werden.
- Aktivitäten-Diagramm (Activity Diagram / act):
  Damit kann das dynamische Verhalten eines Systems mit Hilfe von Aktivitäten beschrieben werden. Dieser Diagrammtyp eignet sich besonders zur Darstellung von Kontroll- und Datenflüssen.
- Zustandsmaschinen-Diagramm (State Machine Diagram – sm)
  Damit kann das dynamische Verhalten eines Systems in Form einer Zustandsmaschine beschrieben werden. Dieser Diagrammtyp eignet sich besonders zur Darstellung von ereignis-basierten Situationen.
- Sequenz-Diagramm (Sequence Diagram – sd):
  Damit kann das dynamische Verhalten eines Systems mit Hilfe von Sequenzen, die den zeitlichen Ablauf darstellen, beschrieben werden. Dieser Diagrammtyp eignet sich besonders zur Darstellung von Interaktionen, die auf den Austausch von Nachrichten basieren.
- Blockdefinitions-Diagramm (Block Definition Diagram – bdd):
  In der SysML werden die Definition eines Systems (egal auf welcher Hierarchiestufe) als Block, und deren Instanzierungen als Komponenten (Parts) bezeichnet. Mit Hilfe eines Block-definitions-Diagramms können statische Zusammenhänge der Blöcke aufgezeigt werden.
- Internes Block-Diagramm (Internal Block Diagram - ibd):
  Dient der Darstellung der internen Verschaltung von Blöcken/Systemen.

Die Notation der SysML kann der Literatur [23] und [24] entnommen werden.

2.5 RODOS

Das „Realtime Onboard Dependable Operating System“ (kurz: RODOS) [6] ist ein vom DLR (Standort Bremen) entwickeltes Framework mit Echtzeitbetriebssystem für eingebettete Systeme. Zu seinen Stärken zählt seine hohe Zuverlässigkeit. Diese wird dadurch erreicht, dass seine Funktionalität mit möglichst geringer Komplexität realisiert wurde. Dadurch bleibt das Framework/Betriebssystem sowohl für den Anwender als auch für zukünftige Weiterentwicklungen, wie sie sowohl beim DLR als auch beim Lehrstuhl Informatik VIII der Universität Würzburg betrieben werden, beherrschbar.

Das Framework ist mehrschichtig modular aufgebaut (siehe Abbildung 2-10) und kann je nach Bedarf individuell zu einem System zusammengesetzt werden.

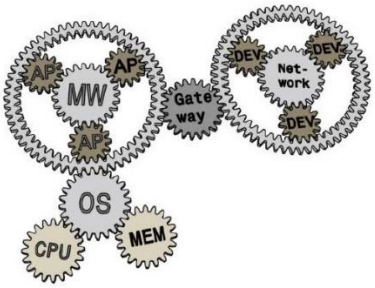


Abbildung 2-10: RODOS und seine Komponenten (Quelle: DLR)

Ein einzelner RODOS-Knoten besteht aus drei Schichten. Auf unterster Ebene befindet sich die Hardwareschicht. In dieser sind Treiber, welche den Zugriff auf die Hardware entkoppeln und eine einheitliche Schnittstelle zur nächst höheren Schicht zur Verfügung stellen, angesiedelt. Die nächsthöhere Schicht beinhaltet das Betriebssystem, die Hauptkomponente des Frameworks. Daran angeschlossen ist eine Middleware samt Netzwerkdiensten zur Kommunikation mit anderen Knoten. In der obersten Schicht sind die Softwareanwendungen platziert. Dank des mehrschichtigen Aufbaus und seiner Abstrahierung können – freie Ressourcen vorausgesetzt - Softwareanwendungen auf beliebigen Knoten platziert werden, da ihnen jeder Knoten immer dieselbe Schnittstelle anbietet.

Auf bestimmte Aspekte von RODOS soll im Folgenden näher eingegangen werden:

Betriebssystem, Scheduler und Threads

Das Betriebssystem ist echtzeitfähig und besitzt einen eingebauten Scheduler zur präemptiven Threadsteuerung. Damit lassen sich unterschiedliche Programmabläufe (aus den Software-anwendungen) in zugeordneten Threads (Programmfäden) scheinbar zur selben Zeit ausführen. Dies wird dadurch erreicht, dass der Scheduler per Zeitmultiplexverfahren (z.B. prioritätsbasiert, oder per Round-Robin) ermittelt, welche Zeitintervalle der CPU einem Thread zur Ausführung bereit stehen. Der Softwareentwickler kann auch programmtechnisch Threads anhalten, abbrechen, fortführen oder sogar miteinander synchronisieren. Des Weiteren besitzt das Betriebssystem eine Interruptverwaltung.

Middleware und Netzwerk

Die Middleware von RODOS dient dem Austausch von Daten zwischen mehreren Teilnehmern (in der Regel Softwareanwendungen, aber auch Hardware-Geräte). Dies geschieht nach dem Publisher/Subscriber-Entwurfsmuster. Der Zugriff auf diesen Mechanismus ist mit Hilfe von Schnittstellen standardisiert. Ein oder mehrere publizierende Teilnehmer können über sogenannte Topics Nachrichten veröffentlichen (siehe Abbildung 2-11). An diesen interessierte Teilnehmer müssen das jeweilige Topic abonnieren.

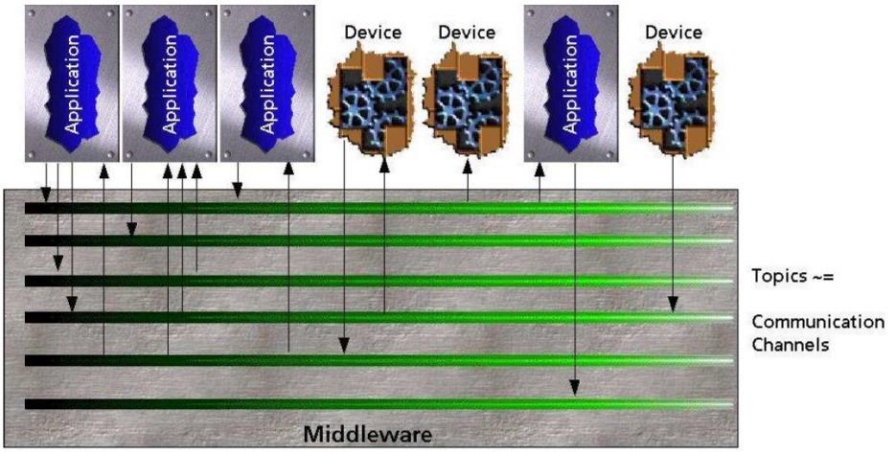


Abbildung 2-11: Datenübertragung mit Hilfe der Middleware (Quelle: DLR)

Mehrere Knoten können zu einem Netzwerk zusammengeschlossen werden. Für den Fall, dass Nachrichten über Knotengrenzen ausgetauscht werden sollen, müssen Gateways eingebunden werden. Diese greifen auf die Dienste von sogenannten Hardware-Abstraction-Layers (HAL) zurück. In diesen ist die Funktionalität zur Datenübertragung per Netzwerkverbindung (z.B. CAN-Bus, Ethernet und TCP/IP) gekapselt.

Die nachfolgende Tabelle zeigt den Aufbau eines Topic-Frames:

Byte-Position	**Name**	**Beschreibung**
0 – 1	Checksum	Die Checksumme des Frames
2 – 5	Sendernode	Die ID des sendenden RODOS-Knoten
6 – 13	Time	Zeitpunkt des Sendens in Nanosekunden
14 – 17	Thread	Die ID des sendenden RODOS-Threads
18 – 21	Topic	Die ID des RODOS-Topics, auf dem das Frame gesendet wird
22 – 23	MaxSteps	Keine Definition
24 – 25	Length	Die Länge des Frames in Bytes
>= 26	Data (Message)	Die Nutzdaten (Nachricht) des Frames

Tabelle 2-1: Der Aufbau eines RODOS-Frames

Plattformen und Portierbarkeit

Es existieren RODOS-Distributionen für verschiedene Plattformen (Prozessor- oder fremde Betriebssysteme). Zu den bekanntesten zählen:

- Linux
- Windows
- Posix
- ARM7
- Xilinx PPC
- Raspberry Pi

Für jede Distribution müssen die Hardware-Treiber und elementaren RODOS-Betriebssystemfunktionen (teils in Assemblersprache) angepasst werden. Im Rahmen der Projekte zur vorliegenden Dissertation wurde das RODOS-Framework für die Xilinx Microblaze Prozessorplattform angepasst.

3 Beispiele autonomer Satelliten

In diesem Kapitel werden einige der bekanntesten Satelliten, die über autonome Funktionalitäten verfügen, vorgestellt. Hierbei stehen zunächst jene Anwendungen, die durch die Autonomie erst ermöglicht wurden, im Vordergrund. Der generelle Aufbau und die Funktionsweise autonomer Planungssysteme hingegen werden im Anschluss in Kapitel 4 (Stand der Technik und der Forschung) beschrieben.

Die Satelliten sind chronologisch nach ihren Starts (Launch) gelistet.

3.1 Solar Maximum Mission

Einer der ersten Satelliten mit autonomen Fähigkeiten [14, p. S.56] war der am 14.02.1980 vom Cape Canaveral Raumhafen gestartete NASA-Satellit SMM der „Solar Maximum Mission“. Seine Aufgabe bestand in der Beobachtung der Sonne. Zu diesem Zweck wurde er mit diversen Sonnensensoren und Spektrometern ausgestattet. Mit Hilfe seines Lageregelungssystems konnte er sich auf bis zu fünf Bogensekunden genau auf sein Ziel ausrichten. Ein zusätzlich implementiertes rudimentäres - zum damaligen Zeitpunkt jedoch technisch weit fortgeschrittenes - autonomes Planungssystem erlaubte es dem Satelliten, Sonnenausbrüche auf der Oberfläche der Sonne zu detektieren. Wenn solch ein Ereignis eintrat, griff das autonome Planungssystem aktiv in die Lageregelung ein und veranlasste, dass der Satellit seine Instrumente zur Beobachtung auf die betroffene Stelle der Sonne ausrichtete.

Abbildung 3-1: SMM Satellit (Quelle: NASA)

3.2 Earth Observing One

Die NASA startete im Jahre 1995 ihr New Millenium Program (NMP) mit dem primären Ziel der Technologieerprobung. Der Satellit EO-1 (Earth Observing One) [25] ist der erste Satellit, der im Rahmen des Technologieprogramms gebaut wurde, und am 21.11.2000 mittels einer Delta 7320 Trägerrakete in einen sonnensynchronen Orbit (705 km) ausgesetzt wurde. Die wissenschaftliche Hauptnutzlast des Erdbeobachtungssatelliten besteht aus einem technisch sehr fortschrittlichen Spektrometer, der sich vielseitig einsetzen lässt: unter anderem für die Erkennung von thermischen Anomalien (z.B. Vulkanausbrüche, Lavaflüsse), von Wolken, von überfluteten Regionen, oder allgemein von Veränderungen von beobachteten Gebieten. Im Rahmen der Technologieerprobung wurde der Satellit ebenfalls mit dem „Autonomous Sciencecraft

Abbildung 3-2: EO-1 Satellit (Quelle: NASA)

Experiment“ (ASE), einem autonomen Planungssystem, ausgestattet. Die Satellitenmission war ein voller Erfolg und bewies die Einsatzreife des ASE, das daraufhin als Grundlage für zukünftige Missionen der NASA verwendet wurde, wenn diese autonome Funktionalitäten benötigen. Das autonome Planungssystem verringerte Arbeitslast und Kosten der Missionsplanung am Boden (Einsparungen von über 1 Mio. $ pro Jahr). Ferner erhöhte es zum einen gravierend die Anzahl wissenschaftlicher Ergebnisse/Aufnahmen, zum anderen reduzierte es die Reaktionszeiten des Satelliten auf mögliche (nichtdeterministische) wissenschaftliche Ereignisse.

3.3 PROBA

PROBA (Project for On-Board Autonomy) ist eine Technologieerprobungsmission der ESA. Der von dem belgischen Unternehmen Verhaert gebaute Satellit wiegt 94 Kg und gehört somit zur Gruppe der Kleinsatelliten. Er wurde am 22.10.2001 in eine ca. 600 Km entfernte Umlaufbahn um die Erde ausgesetzt und wird von einer Bodenstation in Redu (Belgien) gesteuert. Zu seinen Besonderheiten gehört seine selbstständige und autonome Arbeitsweise, die im Rahmen dieser Mission erprobt wurde. Der Satellit verfügt über zwei hochauflösende Kameras zur Erdbeobachtung. Die aufzunehmenden Ziele werden von der Bodenstation lediglich in Form von Höhen- und Breitengraden vorgegeben. Dank seiner autonomen Fähigkeiten kann der Satellit selbstständig berechnen, zu welchen Zeitpunkten Aufnahmen möglich sind, und dementsprechend eigenständig Anweisungen ableiten. Der Satellit gilt als einer der ersten vollautonomen Satelliten, da er seine grundlegenden und sich wiederholenden Aufgaben wie Navigation, Steuerung der Nutzlast und interne Ressourcenverwaltung fast ohne dem Einwirken der Bodenstation plant und ausführt.

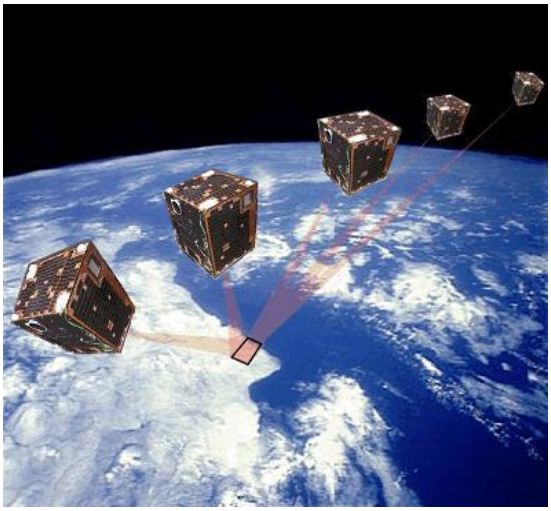


Abbildung 3-3: PROBA Satellit (Quelle: ESA)

3.4 SWIFT

Swift ist ein Forschungssatellit der NASA. Seine Aufgabe besteht in der Detektion und Untersuchung kurzzeitiger Gammablitze. Der Satellit ist 1.331 Kg schwer und wurde am 20.11.2004 von Cape Canaveral aus in eine ca. 600 Km von der Erdoberfläche entfernte Umlaufbahn geschossen. Mit seinem BAT Teleskop (Burst Alarm Teleskop) kann SWIFT Signaturen von Gammablitzen erkennen. Die Flugsoftware ergreift daraufhin autonom folgende Maßnahmen: Zuerst werden die Koordinaten des Ereignisses analysiert und mit bereits erfolgten Aufnahmen verglichen. Im Anschluss wird der aktuell auszuführende Plan angepasst. Dabei werden Kommandos erzeugt, um zum einen die höherauflösenden Instrumente auf das Ziel mittels Lageregelung des Satelliten auszurichten, und zum anderen, um die Instrumente anzusteuern. Die

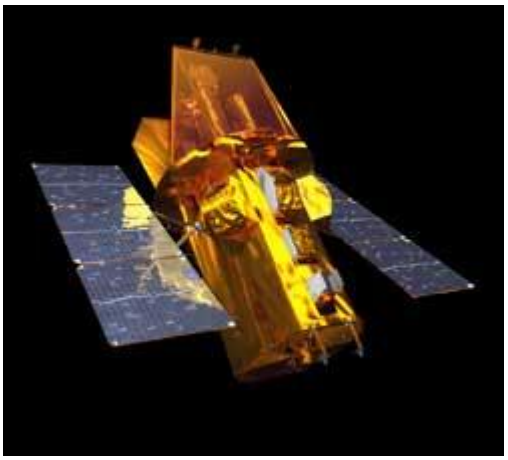


Abbildung 3-4: SWIFT Satellit (Quelle: NASA)

Besonderheit bei SWIFT besteht darin, dass die Steuerung des Satelliten durch die „smarten“ Instrumente anstatt durch Operatoren am Boden übernommen wird.

3.5 XSS 11

XSS 11 (Experimental Satellite System 11) ist ein Technologieerprobungssatellit der US Air Force und wurde vom Air Force Research Laboratory entwickelt. Er wurde am 11.04.2005 in eine ca. 850 Km hohe Umlaufbahn ausgesetzt. Das besondere Merkmal des Satelliten ist seine Fähigkeit, sich autonom anderen Satelliten zu nähern und ein Rendezvous durchzuführen. XSS 11 ist dabei, anders als bei normalen Dockingmanövern, nicht auf ein kooperatives Verhalten des Ziels angewiesen. Diese Fähigkeit ist eine Voraussetzung, um in Anschluss weiterführende Handlungen wie zum Beispiel In-Orbit-Reparaturen und Inspektionen durchzuführen.

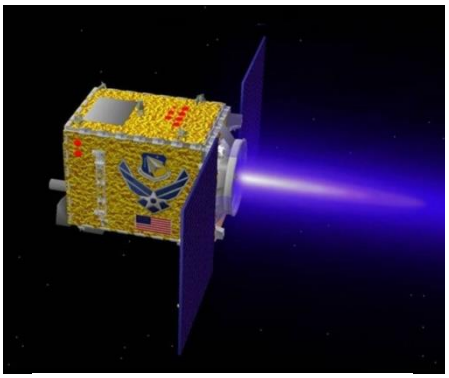


Abbildung 3-5: XSS 11 Satellit (Quelle: ww.space.com)

3.6 PLEIADES

Zwei Erdbeobachtungssatelliten mit fortschrittlichen autonomen Funktionen sind die PLEIADES (1A und 1B) – Satelliten [26] des französisch-italienischen Satelliten-programs ORFEO. Geplant, entwickelt und gebaut wurden die Satelliten unter der Leitung der französischen Raumfahrtagentur (centre national d’etudes spatiales, CNES). Sie wurden am 17.12.2011 (1A) bzw. am 02.12.2012 (1B) mit Hilfe von Sojus-ST Trägerraketen im Orbit ausgesetzt. Zwei herausragende Merkmale der Satelliten sind ihre enorme Beweglichkeit hinsichtlich aller drei Achsen und ihre hochauflösenden Kameras. Mit Hilfe eingebauter autonomer Planungssysteme sind die Satelliten in der Lage, ihre Liste von Zielgebieten selbständig abzufahren und aufzuzeichnen. Die Daten eines zusätzlichen Instruments zur Wolkenerkennung fließen ebenfalls in den autonomen Planungsprozess mit ein. Geplante Aufnahmen von Gebieten, die zum Beobachtungszeitpunkt mit Wolken überzogen sind, werden autonom auf einen späteren Zeitpunkt verschoben. Die Zeit zur autonomen Prozessierung dieser Umstellung, beginnend mit der Wolkendetektion bis zur Änderung der Liste mit den Zielgebieten, beträgt maximal eine Minute und damit deutlich weniger als bei konventioneller Kommandierung.

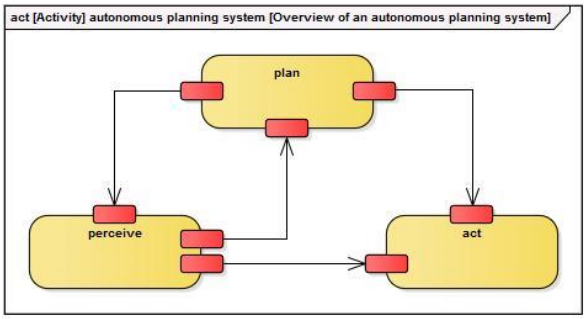
Abbildung 3-6: PLEIADES Satellit (Quelle: CNES)

4 Stand der Technik und der Forschung

Dieses Kapitel soll den aktuellen Stand der Technik und der Forschung im Bereich autonomer Satelliten aufzeigen. Zu Beginn werden allgemeine Aspekte und Begriffe bzgl. der Autonomie, im Anschluss daran einzelne autonome Planungssysteme vorgestellt.

4.1 Allgemeine Definitionen von autonomen Planungssystemen

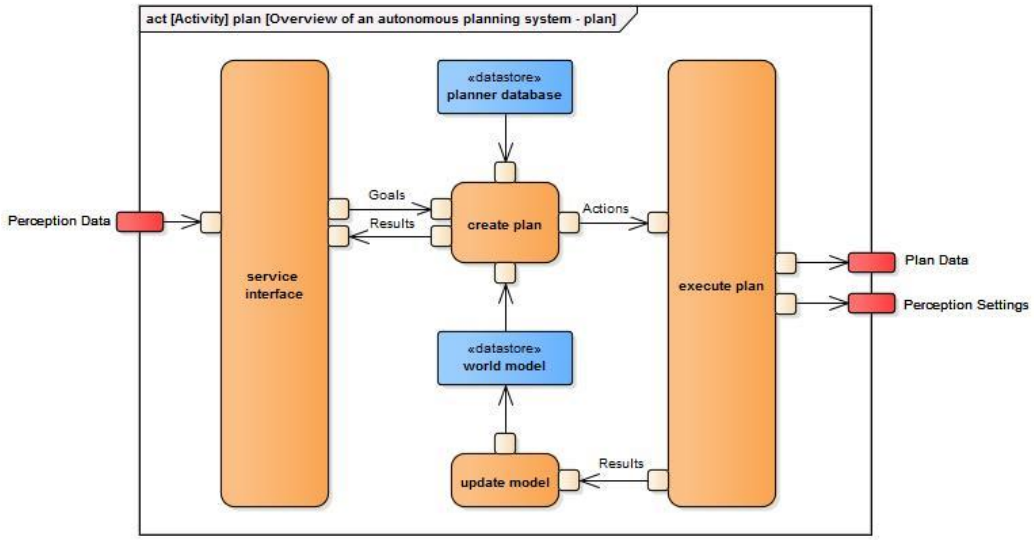
Generell muss ein autonomes Planungssystem zur Erreichung seines Zwecks drei Aufgaben (siehe Abbildung 4-1) erfüllen.



In Anlehnung an: [15, p. S.96], überarbeitet

Abbildung 4-1: Aufgaben eines autonomen Planungssystems

Erstens muss ein autonomes Planungssystem seine Umgebung wahrnehmen können, zweitens muss es seine Umgebung auch beeinflussen können, und drittens muss es in der Lage sein, Planungen durchzuführen. Für die erste Aufgabe können sowohl Daten über Sensoren erfasst werden, als auch über Kommunikationswege mitgeteilt werden. Da die Daten meist aus mehreren Quellen stammen, müssen sie mittels geeigneter Methoden fusioniert werden. Die Auswertung der Daten kann zu direkten Reaktionen des autonomen Planungssytems führen (niedere Autonomie). Sie kann auch der Planung zufließen und somit Auswirkungen auf die zukünftigen Handlungen des Systems haben (höhere Autonomie). Die eigentliche Planung wird in Abbildung 4-2 dargestellt.



In Anlehnung an: [14, p. S.97], überarbeitet und modifiziert

Abbildung 4-2: Aktionen beim Planen

Sie greift als Basis für ihre Entscheidungen in der Regel auf ein Modell der Umgebung und auf eine Datenbank mit ihren Verhaltensmustern, wie z.B. Planungszielen, zurück. Diese können über eine Service-Schnittstelle eingepflegt werden. Der erstellte Plan wird ausgeführt, und dabei entstehende Resultate werden in das Weltmodell eingepflegt.

Der Entwurf autonomer Planungssysteme orientiert sich meistens an zwei Programmierparadigmen: Niedere autonome Funktionalitäten werden gewöhnlich, auf Grund der üblicherweise geforderten kurzen Reaktionszeiten, mit Hilfe des ereignisgesteuerten Programmierparadigma erstellt. Der Aufbau höherer autonomer Funktionalitäten orientiert sich hingegen mehr an das zielorientierte Programmier-paradigma.

Die ECSS behandelt das Thema Autonomie im Standard ECSS-E-ST-70-11C [27, pp. S.32-39]. Die Anwendungen der Autonomie werden hierin in die drei (Anwendungs-)Klassen, Missionsausführung (engl. mission execution), Speicherverwaltung (engl. data management) und Fehlerbehandlung (FDIR) aufgeteilt und je nach Grad der Autonomie in verschiedene Levels eingeteilt. Die Missionsausführung unterscheidet die Levels E1 (niedrig) bis E4 (hoch). E1 und E2 entsprechen von der Funktionalität her den Möglichkeiten eines konventionellen Satelliten. E3 ist mit einem mit niederen autonomen Planungssystem (ereignisorientiert), E4 mit einem mit höheren autonomen Planungssystem (ereignis- und zielorientiert) ausgestatteten Satelliten vergleichbar. Speicherverwaltung und Fehlerbehandlung werden jeweils getrennt in die Levels D1 (niedrig) und D2 (hoch) unterschieden.

Des Weiteren grenzt die Literatur autonome Funktionalitäten daran voneinander ab, ob sie neue Anwendungen z.B. erst ermöglichen (engl. enabling technology), oder ob sie lediglich bereits bestehende Prozesse verbessern (engl. improvement technology). Verbesserungen könnten in diesem Sinn unter anderem Arbeitserleichterungen für die Belegschaft oder Kostenersparnisse sein.

4.2 Softwareagenten

Eine häufig eingesetzte Technologie zur Realisierung von autonomen Systemen ist die der Software-Agenten. Wie im Fall der Autonomie ist der Begriff „Software-Agent“ in der Literatur nicht eindeutig definiert. Agenten sind Software-Entitäten (z.B. Klassen in objekt-orientierten Programmiersprachen), die autonome Funktionalitäten kapseln und diese über eine Schnittstelle zur Verfügung stellen. Sie unterscheiden sich untereinander vor allem in ihrem Verwendungszweck und ihrem Einsatzgebiet. Dementsprechend sind ihre Fähigkeiten und Eigenschaften unterschiedlich stark ausgeprägt. Die nachfolgende Tabelle stellt eine grobe Übersicht der wichtigsten Eigenschaften von Software-Agenten dar.

Eigenschaft	**Beschreibung**
kognitiv	Der Agent ist lernfähig und wertet seine Auswirkungen auf die Umwelt aus. Daraus gewonnene Erfahrungen fließen in den Entscheidungsprozess mit ein.
reaktiv	Der Agent reagiert auf externe Ereignisse.
initiativ	Der Agent agiert nach eigenem Ermessen.
echtzeitfähig	Der Agent braucht maximal eine bestimmte Zeitspanne, um zu reagieren und/oder zu agieren.
reflexiv	Der Agent analysiert seine eigenen Handlungen hinsichtlich bestimmter Kriterien (z.B. Fehler, Erfolg, Verhalten, …).
parametrierbar	Das Verhalten und die Fähigkeiten des Agenten sind anpassbar.
kommunikativ	Der Agent kann mit anderen Softwarekomponenten (meistens andere Agenten) kommunizieren. Der Datenaustausch ist durch Protokolle und/oder eine Sprache geregelt.
adaptiv	Der Agent kann sein Verhalten durch selbständiges Ändern seiner Parameter verändern.

Tabelle 4-1: Eigenschaften von Software-Agenten

Besonders bei sehr komplexen Problemen wird eine Hierarchie von Softwareagenten eingesetzt. Dabei wird das Problem in Teilprobleme zerlegt und jeweils einzelnen speziell für die Lösung des Teilproblems geeigneten Agenten überantwortet. Ein übergeordneter Agent übernimmt Kontrollaufgaben und delegiert Teilaufgaben an seine untergeordneten Agenten. Die Kommunikation zwischen zwei oder mehreren Agenten ist meistens in einer Sprache definiert.

4.3 Framesworks: ASPEN und CASPER

ASPEN (Automated Scheduling and Planning Environment) [28], [29] ist ein fortschrittliches Software-Framework, das zur Erstellung von Anwendungen sowohl für das Bodensegment als auch für das Raumsegment verwendet werden kann. Es wurde vom JPL (Jet Propulsion Laboratory) des California Institute of Technology entwickelt. Der Schwerpunkt des Frameworks liegt, wie sein Name schon andeutet, in der Erstellung von Anwendungen zur Planung und zur Ablaufsteuerung. Der planenden Komponente können Ziele vorgegeben werden, von denen durchzuführende Aktivitäten abgeleitet werden. Die Ablaufsteuerung nimmt diese Aktivitäten als Ausgangspunkt und erstellt eine Kommandosequenz. Sowohl bei der Planung als auch bei der Ablaufsteuerung können Beschränkungen vorgegeben werden, die zum Beispiel die Aktivitäten oder die Ressourcen betreffen.

Das Framework wurde nach dem objektorientierten Paradigma entworfen und hat daher als Grundlage ein Klassenmodell. Dieses besteht aus sieben Hauptklassen, mit denen die Anwendung für die Ablaufsteuerung modelliert werden kann, darunter solche für Aktivitäten, Parameter, Abhängigkeiten, Beschränkungen, Reservierungen, Ressourcen und Zustandsvariablen. Die Instanzen der Klassen werden an zentralen Stellen in der Anwendung verwaltet. Dazu gehören die Datenbank für die Aktivitäten, ein Netzwerk für zeitliche Beschränkungen, Listen von Zeitlinien für die Ressourcen und die Zustände, ein Netzwerk für die Abhängigkeiten zwischen den Parametern, und die Instanz, welche die eigentlichen Planungs- und Ablaufsteueralgorithmen enthält. Der Anwender des Frameworks kann zur Unterstützung und zur einfacheren Bedienung eine graphische Oberfläche verwenden.

Derzeit sind im Framework drei Algorithmen für die Planung und die Ablaufsteuerung implementiert. Sie besitzen folgende Charakteristiken (in Englisch):

- forward dispatch, greedy, constructive
- backtracking, constructive (IRS-Algorithmus)
- iterative repair (DCAPS-Algorithmus)

Je nach Anwendungsdomäne kann der Anwender des Frameworks einen passenden Algorithmus für sein autonomes Planungssystem aussuchen.

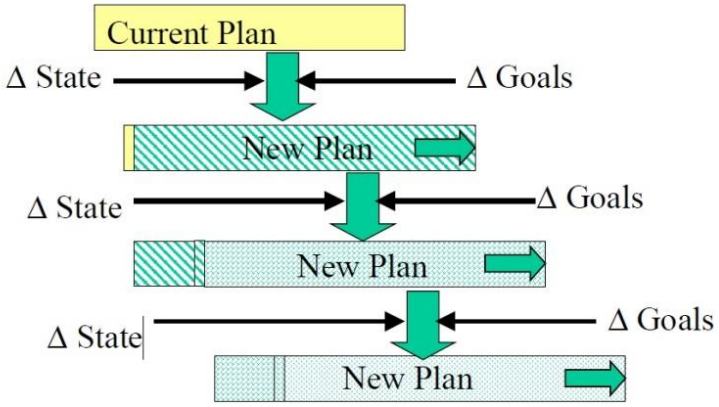


Abbildung 4-3: Kontinuierliches Planen und inkrementelles Erweitern des Plans (Quelle: [29])

Eine Variante des ASPEN-Frameworks ist auch unter dem Namen CASPER [30] (Continuous Activity Scheduling Planning Execution and Replanning) bekannt. Diese wurde gegenüber dem ASPEN um Echtzeitfähigkeiten erweitert, damit sich die erstellten Anwendungen besser für dynamische Planungssituationen eignen. Der wesentliche Unterschied zu ASPEN liegt in der Verwendung eines kontinuierlich planenden und inkrementell erweiternden Algorithmus (siehe Abbildung 4-3) für die Erstellung von Plänen (betrifft sowohl die Planung als auch die Ablaufsteuerung).

Das Wesensmerkmal [31], [32] des Algorithmus liegt darin, dass die Kommandosequenz (der Plan) für den Satelliten kontinuierlich mit dem Fortschreiten der Zeit oder dem Eintreten von Ereignissen angepasst wird. Der Plan gilt dabei nur für eine bestimmte Zeitspanne (dem Planungshorizont). Bei den fortlaufenden Anpassungen des Plans kann es passieren, dass Konflikte (z.B. Ressourcenmangel) auftreten. In einem solchen Fall wird der Plan ungültig. Zur Behebung des invaliden Plans kann der Algorithmus auf verschiedene Reparaturmethoden (z.B. zeitliche Verschiebung der Aktivitäten) zurückgreifen. Dabei können vom Anwender bestimmte Heuristiken vorgegeben werden, um den Suchraum des Algorithmus für die Lösungsfindung einzuschränken. Des Weiteren pflegt der Algorithmus eine Hierarchie von Planungshorizonten mit unterschiedlicher Dauer. Prinzipiell gilt, je länger der jeweilige Horizont, desto abstrakter seine Planung. Das bedeutet, dass zum Beispiel der längste Horizont lediglich aus abstrakten Aktivitäten besteht. Solch eine Aktivität würde zum Beispiel einer Flugprozedur am Boden entsprechen. Je kürzer der Horizont, desto konkreter werden die Aktivitäten. Im Grunde findet eine Zerlegung statt, bis beim kürzesten Horizont jeder Aktivität ein Kommando des Satelliten zugeordnet werden kann.

ASPEN und CASPER wurden in verschiedenen Satelliten-Missionen eingesetzt. Dazu zählen unter anderem:

- Citizen Explorer (CX-1)
- Antarctic Mapping Mission (AMM-2)
- Earth Observer One (EO-1)
- UHF Follow On One (UFO-1)

Obwohl die beiden Frameworks sehr fortschrittlich sind, reichen sie alleine für eine Satellitenmission nicht aus, sondern dienen lediglich als Grundlage für entsprechende Anwendungen. Ein mögliches Beispiel wird im nächsten Abschnitt vorgestellt.

4.4 Autonomous Sciencecraft Experiment

Das Experiment „Autonomous Sciencecraft Experiment“ (ASE) [33], [34], [35] verfolgt das Ziel, einen Software-Agenten, der einen Satelliten autonom steuert, zu entwickeln und anschließend im Orbit zu testen. Ursprünglich sollte das Experiment auf den Techsat-21 [36] Satelliten der NASA durchgeführt werden, doch diese Satelliten-Mission wurde aus finanziellen Gründen abgebrochen. Letztendlich wurde das Experiment am NASA-Satelliten EO-1 [25] erfolgreich durchgeführt. Der Agent hatte dadurch Einsatzreife bewiesen und gilt bei der NASA heute noch als Muster für zukünftige Satelliten-Missionen, die auf autonome Funktionalitäten angewiesen sind.

Die nachfolgende Abbildung 4-4 zeigt die grundlegende Architektur des Agenten. Sie baut auf dem im Abschnitt 4.2 vorgestellten CASPER-Framework auf.

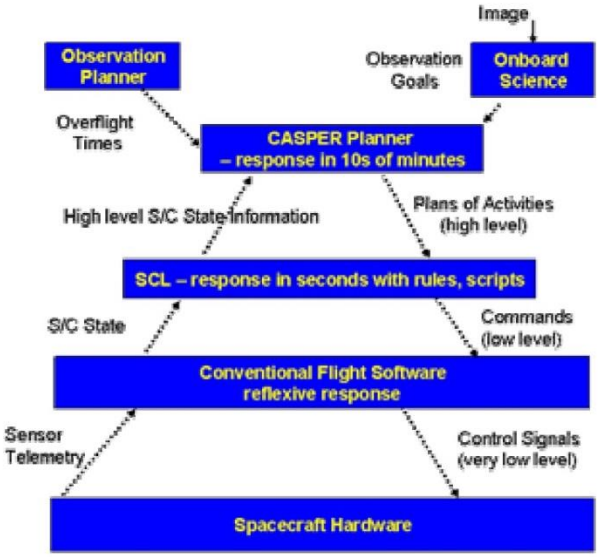


Abbildung 4-4: Architektur des Software-Agenten für das ASE (Quelle: [33])

ASE baut auf drei Softwareschichten auf: Auf der obersten Schicht steht der Planer des CASPER-Framesworks. Dieser ist für die Missionsplanung des Satelliten verantwortlich und wird zum einen über Ereignisse angestoßen. Diese werden z.B. beim Satelliten EO-1 vom Software-Modul, das wissenschaftliche Auswertungen vornimmt, ausgelöst. Beispiele sind das Detektieren von Vulkanausbrüchen oder Überschwemmungen mit Hilfe seiner Multispektralinstrumente. Zum anderen arbeitet der Planer, wie bereits im vorherigen Abschnitt erwähnt wurde, auch zielorientiert. Ein sehr wichtiges Ziel ist zum Beispiel der Aufbau einer Funkverbindung zur Bodenstation, falls die Möglichkeit eines Funkkontakts besteht. Die dazu notwendige Information, wann solche Gelegenheiten gegeben sind, erhält der CASPER-Planer vom Flug-Planer. Des Weiteren teilt der Flug-Planer dem CASPER-Planer auch mit, wann sich besonders günstige Gelegenheiten für wissenschaftliche Beobachtungen ergeben. Mit diesen Informationen kann der CASPER-Planer die nötigen Aktivitäten für den Satelliten ableiten und anschließend einen passenden Plan erstellen, der dann als Skript an die beiden Software-Module der mittleren Schicht zur Ausführung übergeben wird. Bei den besagten Modulen handelt es sich zum einen um das SCL (Spacecraft Command Language) [37], das regelbasiert und prozedural arbeitet, und zum anderen um das MI-R (Model-based mode identification and execution), das deklarativ stochastisch mit einem finiten Zustandsmodell arbeitet. SCL eignet sich aufgrund seiner Regel-Engine besonders für autonome Reaktionen in Echtzeit, d.h. immer dann, wenn schneller Handlungsbedarf besteht und Ergebnisse bis zu einem definiertem Zeitpunkt vorliegen müssen. Deswegen sind auch große Teile der FDIR-Funktionalität als SCL-Skripts hinterlegt. Des Weiteren werden mit Hilfe von SCL-Regeln Beschränkungen (engl. constraints) zur Ausführung sowohl von Kommandos als auch von Funktionen hinterlegt. Somit wird jede vom Satelliten auszuführende Aktion, egal ob diese autonom geplant wurde oder per Telekommando initiiert wurde, gegen ihre Bedingungen geprüft und nur im positiven Fall ausgeführt. Durch diese Maßnahme wird die Sicherheit des Satelliten gravierend erhöht. MI-R arbeitet hingegen, wie bereits erwähnt wurde, deklarativ mit einem Zustandsmodell. In diesem Fall gibt der CASPER-Planer lediglich vor, in welchen Zustand eine Komponente (sei es Hardware oder Software) des Satelliten versetzt werden soll. MI-R leitet dann eigenständig mit Hilfe seines eingepflegten Zustandsmodells die dafür notwendigen Aktionen (Kommandos oder Funktionsaufrufe) für den gewünschten Zustandsübergang ab, falls nötig, bei Zwischenzuständen sogar über Alternativpfade (Redundanzen). SCL und MI-R sollen dabei als komplementäre Software-Module gesehen werden, die sich gegenseitig ergänzen. Auf der untersten Softwareschicht existiert die konventionelle Flugsoftware des Satelliten, die letztendlich die geplanten Aktionen ausführt.

4.5 Reaktive/Deliberative - Architektur

Einen generischen Ansatz für ein autonomes Planungssystem stellt die Reaktive/Deliverative-Architektur [38], [39] dar. Sie kommt beim hochagilen französischen Erdbeobachtungssatelliten PLEAIDES (siehe Abschnitt 3.6), insbesondere bei der Planung der Abfolge von Erdaufnahmen, zum Einsatz. Abbildung 4-5 zeigt den strukturellen Aufbau dieser Architektur.

Abbildung 4-5: Reaktive/Deliverative – Architektur (Quelle: [38])

Sie besteht aus zwei Hauptkomponenten, zum einen dem Reaktive Control Task (RCT) und zum anderem den Deliberative Reasoning Task (DRT). Beide Komponenten enthalten autonome Funktionalitäten, die sich jedoch in ihrer Art und Weise unterscheiden. Der RCT ist derjenige von beiden, der an die Schnittstelle der Umwelt (engl. Environment) angebunden ist. Er ist in der Lage, sowohl von dieser Stimuli in Form von Änderungsbenachrichtigungen über den Zustand der Umwelt zu bekommen als auch diese direkt zu beeinflussen. Die Handlungen des RCT sind reaktiver Natur und unterliegen starken zeitlichen Beschränkungen. Aus diesem Grund sind die Berechnungen des RCTs möglichst kurz und bestehen aus einfachen Anweisungen. In diesem Sinne ist der RCT der niederen Autonomie zuzuordnen. Er hat zudem die Aufgabe, den DRT zu kontrollieren, ihm für seine Planung notwendige Informationen weiterzureichen und sich von ihm Pläne erstellen zu lassen. Die Pläne beschreiben autonome Handlungen (höhere Autonomie) und sollten vom RCT lediglich als eine Empfehlung behandelt werden. Es steht dem RCT frei, sich mehrere Pläne berechnen zu lassen, um anschließend den geeignetsten (oder gar keinen) auszuwählen. Auf jeden Fall trägt der RCT die Verantwortung für das gesamte System und muss auch in der Lage sein, mit der Umwelt zu agieren, wenn keine Pläne vom DRT vorliegen.

Dieser Ansatz gibt lediglich die statische Struktur der Software vor. Die dynamischen Abläufe, insbesondere die Planungsalgorithmen, sind wegen des generischen Charakters dieser Architektur nicht genau definiert und müssen je nach Anwendung bzw. Satellitenmission expliziert erstellt werden. Der nächste Abschnitt geht daher exemplarisch auf Überlegungen zu Algorithmen für die Planung von Erdbeobachtungsmissionen, wie sie beim Satelliten PLEAIDES vorkommen, ein.

4.6 Ablaufplanungsalgorithmen für Erdbeobachtungen

Das Ziel einer Ablaufplanung für Erdbeobachtungen ist es, eine Sequenz von Aufnahmen zu berechnen, die mit der Kamera des Satelliten auch tatsächlich durchgeführt werden kann. Abbildung 4-6 veranschaulicht diese Thematik.

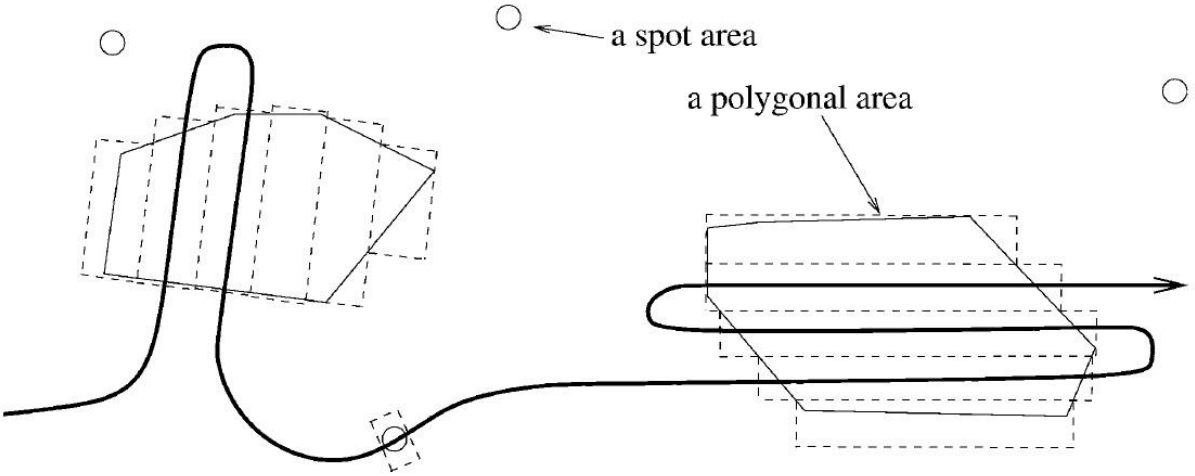


Abbildung 4-6: Beispiel einer geplanten Spur einer Kamera an Bord eines Satelliten (Quelle: [32])

Die Ablaufplanung ist ein Optimierungsproblem, da es im Interesse des Satellitenbetreibers liegt, seine geplanten Aufnahmen hinsichtlich eines Ziels (z.B. so viele Aufnahmen wie möglich) zu optimieren. Im Rahmen der Forschungsarbeiten zum Satelliten PLEAIDES wurden verschiedene Algorithmen [40] untersucht, mit denen sich Lösungen zum Ablaufplanungsproblem finden lassen. Dies schloss die mathematische Modellierung des Problems mit ein.

Ausgangspunkt der Modellierung sind die vom Kunden in Auftrag gegebenen, mit Hilfe der Kamera des Satelliten anzufertigenden, Aufzeichnungen von Gebieten auf der Erde. Diese werden, wenn nötig, in mehrere parallel anliegende Abschnitte, die kompatibel zur Spatbreite der Kamera sind (siehe Abbildung 4-6), aufgeteilt und als Auftragsinput dem Planungsalgorithmus zugestellt. Zu jeder Aufnahme lassen sich folgende Attribute modellieren:

- frühestmöglicher Startzeitpunkt der Aufnahme
- spätmöglichster Startzeitpunkt der Aufnahme
- Dauer der Aufnahme
- Gebietsgröße der Aufnahme am Boden

Des Weiteren werden im Modell alle Zeiten angeben, die der Satellit für die Neuausrichtung zwischen zwei möglichen Aufnahmen benötigt.

Eine zusätzliche Anzahl an Beschränkungen im Modell begrenzt den Suchraum von möglichen Lösungen für die Algorithmen:

- Der durch den Algorithmus zugewiesene Startzeitpunkt einer Aufnahme muss zwischen dem frühestmöglichen und dem spätestmöglichen liegen.
- Die Aufnahmen dürfen sich zeitlich nicht überlappen.

Zuletzt muss im Modell noch eine Zielfunktion angeben werden, die die Algorithmen veranlasst, im Rahmen ihrer Optimierung den Zielwert zu maximieren. Eine einfache Möglichkeit hierfür wäre zum Beispiel, alle Aufnahmen mit einer Gewichtung zu versehen, die den einzelnen Einnahmen der Aufzeichnungen entspricht.

Folgende angepasste Varianten von Standard-Algorithmen könnten zur Lösungsfindung des Problems angewendet werden:

- Dynamische-Programmierung-Algorithmus (DPA)
- Greedy-Algorithmus (GA)
- Constraint-Programming-Algorithmus (CPA)
- Lokale-Suche-Algorithmus (LSA)

Der DPA betrachtet das Problem graphen-theoretisch. Jeder Knoten des Graphen entspricht einer Aufnahme und jede Kante einer Nachfolgerbeziehung zwischen zwei Aufnahmen. Der Knoten wird zusätzlich mit dem individuellen Gewinn der Aufnahme gewichtet. Jede Kante erfüllt die zuvor modellierten Beschränkungen. Der DPA versucht, den am stärksten gewichteten durchführbaren Pfad im Graphen zu finden, um den Gewinn zu maximieren. Der Pfad entspricht einer zeitlich geordneten Folge von durchzuführenden Aufnahmen. Charakteristisch für diesen Typ von Algorithmus ist die Tatsache, dass berechnete Teilpfade zwischengespeichert und wiederverwendet werden, um eine erneute Berechnung zu vermeiden (Dynamische Programmierung).

Der GA baut sequentiell eine Folge von Aufnahmen als Lösung auf. Als Grundlage verwendet er eine nach Startzeitpunkten geordnete Liste aller möglichen Aufnahmen. Jeder Aufnahme in der Liste ist ein eigener Gewinn sowie ein möglicher nachfolgender Gewinn, der sich aus der restlichen verfügbaren Planungszeit ergibt, zugeordnet. Der GA untersucht sequentiell die Liste, ob eine Aufnahme in die Lösungsmenge aufgenommen werden soll. Die einzelnen Gewinne in der Lösungsmenge werden summiert. Der „gierige“ Charakter des GA besteht darin, dass er die Aufnahmen mit einem höheren möglichen Folgegewinn bevorzugt.

Beim CPA-Ansatz wurde der Algorithmus nicht selbst entworfen, sondern das Problem in der Optimization Programming Language (OPL), bestehend aus einem Framework und einer Sprache, modelliert. Dies erfolgt in Form von Daten, Beschränkungen, Entscheidungsvariablen und Optimierungskriterien. Ein dem Framework beigefügter Solver löst das modellierte Problem systematisch. Da der Suchraum an möglichen Lösungen meistens zu groß ist, muss er mit weiteren Beschränkungen künstlich eingeschränkt werden, um in akzeptablen Zeiten Lösungen zu erhalten.

Der LSA gehört zur Gruppe der suboptimalen Algorithmen, das heißt, er kann nicht garantieren, die beste mögliche Lösung zu finden. Seine Funktionsweise ist iterativ, indem er, ausgehend von seiner aktuellen Lösung, versucht, in seinem Umfeld eine bessere Lösung zu finden. Dies geschieht in der Weise, dass kleinere Ausschnitte der Lösung manipuliert werden, z.B. durch das Hinzufügen, Entfernen oder Verschieben von Aufnahmen. Heuristiken bestimmen je nach Situation, welche dieser Aktionen anzuwenden sind. Sie dienen somit als Richtlinie für das Verhalten des Algorithmus. Den Heuristiken wurde ein Zufallsfaktor beigefügt, um das Verhalten des Algorithmus flexibler zu gestalten.

4.7 ADEPT - Framework

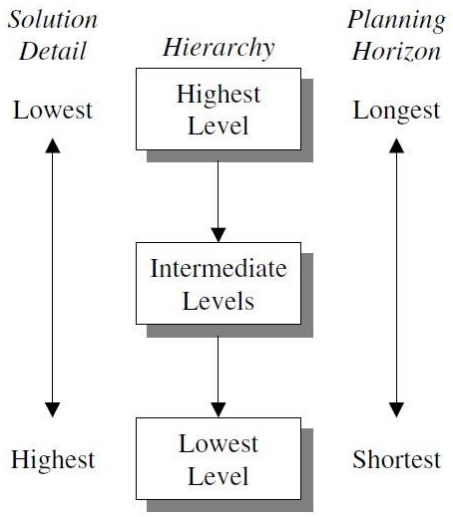


Abbildung 4-7: Hierarchien im ADEPT – Framework (Quelle: [41])

Das ADEPT–Framework [41], [42], [43] eignet sich für die Erstellung von autonom arbeitenden Anwendungen, die für ein hochdynamisches Umfeld optimiert sind. Der Problematik mit der Dynamik und der damit zwangsläufig einhergehenden Ungültigkeit autonom erstellter Pläne wird mit einer Plan-Hierarchie entgegen gewirkt. Abbildung 4-7 verdeutlicht diesen Sachverhalt.

Der für die autonomen Handlungen benötigte Plan wird in unterschiedlichen Hierarchiestufen mit sich unterscheidenden zeitlichen Planungshorizonten erstellt. Pläne auf höheren Hierarchiestufen haben einen größeren Planungshorizont, sind jedoch weniger detailliert. Umgekehrt, haben Pläne auf niedrigeren Hierarchiestufen einen zeitlich beschränkteren Horizont, sind jedoch detaillierter. Somit werden die für die Planberechnung benötigten Ressourcen schonender eingesetzt, da davon auszugehen ist, dass Pläne in einem hochdynamischen Umfeld auf Grund externer Ereignisse ihre Gültigkeit verlieren können. Der Hauptgedanke besteht darin, detaillierte Pläne erst kurz vor der Ausführung zu berechnen.

Das Framework sieht vor, dass auf jeder Hierarchiestufe zusätzlich zur Planerstellung weitere Funktionen ausgeführt werden. Diese werden durch separate Module gekapselt, die zusammen in eine geordnete Interaktion treten. Abbildung 4-8 zeigt den Datenfluss zwischen den einzelnen Modulen. Jede Hierarchiestufe kann als ein Teilsystem des Gesamtsystems aufgefasst werden.

Für die Einschätzung der Lage sind zwei Module verantwortlich. Beim ersten handelt es sich um das Monitoring-Modul. Dieses sammelt Daten (z.B. Signale oder Zustände) über andere Komponenten des zu kontrollierenden Systems, wie Sensoren, Aktoren oder andere Software-Komponenten. Das Monitoring-Modul gleicht die Daten mit Hilfe seiner eigenen Modelle des Systems (Satellit) ab und stellt Abweichungen von Erwartungswerten fest. Bei solchen Abweichungen könnte es sich zum Beispiel um mögliche Fehler im System handeln oder um neue Ereignisse, auf die es, zwecks wissenschaftlicher Untersuchung, zu reagieren gilt.

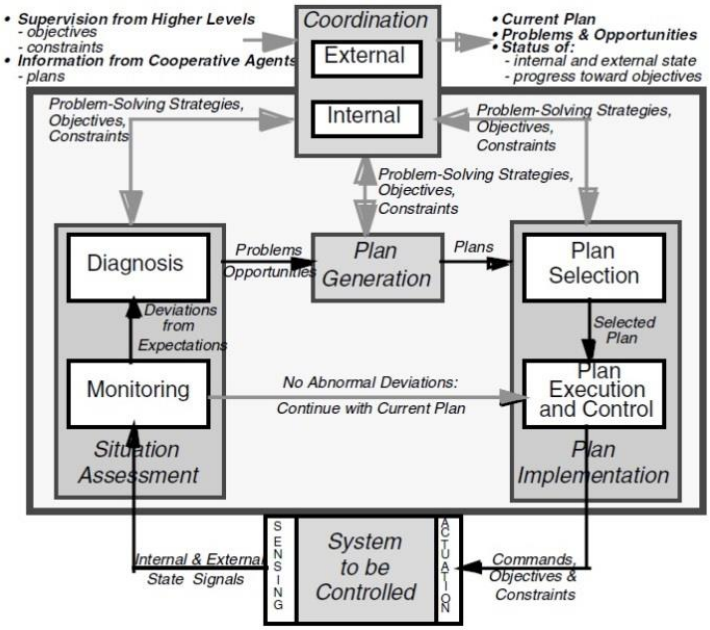


Abbildung 4-8: funktionale Zerlegung im ADEPT – Framework (Quelle: [41])

Das zweite Modul zur Lageeinschätzung ist das Diagnose-Modul. Dieses Modul analysiert die zuvor durch das Monitoring-Modul festgestellten Abweichungen und versucht dabei, zum einen deren Ursache und zum anderen deren Auswirkungen auf das System festzustellen. Das Diagnose-Modul hat als oberste Priorität, die Einsatzfähigkeit der Hierarchiestufe mit Hilfe der anderen Software-Module sicherzustellen. Dazu ist es ermächtigt, Rekonfigurationen durchzuführen, um Fehlerfälle zu vermeiden.

Die Module zur Einschätzung der Lage liefern den Input für das Plan Generation Modul. Dieses ist das Modul zur Erstellung der Pläne. Beim Input handelt es sich in erster Linie um eine Beschreibung der diagnostizierten Probleme und/oder um erkannte Ereignisse, auf die es zu reagieren gilt. Die Verarbeitung des Input bewirkt meistens, dass der aktuelle Plan angepasst werden muss. Dabei können mehrere Planungsalgorithmen zum Einsatz kommen, von denen jeder einen eigenen Plan als Vorschlag unterbreitet. Ein Algorithmus mag zum Beispiel heuristischen Charakter aufweisen, um möglichst schnell einen Plan (z.B. für eine kritische Fehlerbehandlung) zu generieren, während ein anderer Algorithmus eine gründliche Tiefensuche durchführt, um einen optimaleren Plan zu generieren.

Für die Umsetzung der Pläne sind zwei Software-Module verantwortlich. Das erste Modul trifft eine passende Auswahl zwischen den verschiedenen vorgeschlagenen Plänen, während das zweite Modul den selektierten Plan am zu kontrollierenden System ausführt.

Bei einer höheren Hierarchiestufe handelt es sich bei dem zu kontrollierenden System um die nächstniedere Hierarchiestufe. Erst auf der niedrigsten Hierarchiestufe wird das eigentliche zu kontrollierende System direkt über Aktoren angesteuert und über Sensoren erfasst. Die Verbindung zwischen zwei benachbarten Hierarchiestufen erfolgt über ein Coordination-Modul. Dessen Hauptaufgabe ist es, den erstellten Plan der höheren Hierarchiestufe an die niedere Hierarchiestufe weiterzureichen und deren Software-Module bei deren Ausführungen zu unterstützen.

Das Paper [42] beschreibt den Einsatz des ADEPT-Frameworks im Earth Phenomena Observing System (EPOS). Ausgangspunkt in diesem (vorläufig fiktiven) EPOS-Szenario der NASA ist eine Anzahl (im zweistelligen Bereich) von Kleinsatelliten, die als einzige Aufgabe haben, Phänomene auf der Erde zu detektieren. Die Idee dahinter besteht darin, diese Aufgabe aus Kostengründen statt mit einem großen Erdbeobachtungssatelliten mit vielen einfachen Kleinsatelliten durchzuführen. Wegen der großen Anzahl an Kleinsatelliten ist es notwendig, dass sie ihre Planung und ihren Betrieb autonom durchführen. Als Grundlage hierfür soll ADEPT dienen.

Für diese Mission soll ein ADEPT–System mit sieben Hierarchiestufen zum Einsatz kommen:

1. Die erste und abstrakteste Stufe, d. h. die mit dem längsten und gröbsten Planungshorizont, soll bestimmen, welche Bereiche auf der Erde beobachtet werden sollen, und welche Satelliten dafür in Frage kommen.
2. Die nächste Stufe soll zum einen festlegen, ob bereits im Orbit vorhandene Satelliten wiederbetankt werden sollen (eine Technologie, die für das Missions-Szenario als gegeben vorausgesetzt wurde), und zum anderen, ob neue Satelliten in den Orbit geschossen werden müssen.
3. Die dritte Stufe bindet endgültig einen Beobachtungsbereich an einen Satelliten.
4. Die vierte Stufe legt für jeden Satelliten die Auswahl der verwendeten Instrumente und Sensoren fest.
5. Die fünfte Stufe bestimmt die dafür notwendigen Manöver und berechnet die dafür notwendigen Geschwindigkeitsdifferenzen.
6. Die sechste Stufe legt für jedes verwendete Instrument den Zeitpunkt und die Ausrichtung fest.
7. Die siebte Stufe legt für jeden Satelliten fest, welche Nutzdaten gespeichert werden sollen, und wie sie zur Bodenstation übertragen werden sollen.

4.8 AGATA

AGATA [44], [45] ist ein weiteres von der französischen Raumfahrtagentur (CNES) gefördertes Projekt. Dieses setzt im besonderen Maße Generik ein. Die Entwickler erhoffen sich, eine Architektur zu konzipieren, die hochgradig flexibel einsetzbar ist. Sie soll sich für verschiedene Weltraum-

missionstypen, zum Beispiel zur Erdbeobachtung, zur Überwachung oder Erkundung, eignen. Der Hauptansatz hierfür liegt in der Modularität. Das zu kontrollierende Gesamtsystem (d.h. der Satellit) wird in eine Hierarchie von Teilsystemen zerlegt, wobei jedes durch ein eigenes generisches Kontroll-Modul gesteuert wird. Die nachfolgende Abbildung 4-9 zeigt den prinzipiellen Aufbau eines solchen Kontroll-Moduls.

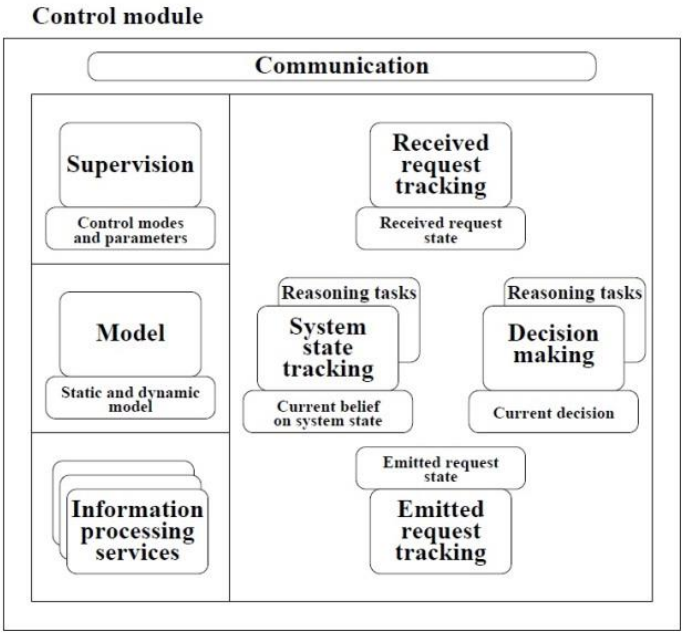


Abbildung 4-9: Aufbau eines generischen Kontroll-Moduls (Quelle: [44])

Ein Kontroll-Modul besteht insgesamt aus sieben Komponenten, von denen vier jeweils eine Standardaufgabe bei der Kontrollausübung übernehmen:

1. das Empfangen und Verwalten von Kontrollanfragen von anderen Kontroll-Modulen
2. das Senden von Kontrollanfragen an andere Kontroll-Module
3. das Erfassen des aktuellen Zustandes des zu kontrollierenden Systems
4. die eigentliche Entscheidungsfindung über ausgehende Kontrollanfragen unter Einbeziehung der Daten/Informationen sämtlicher Komponenten im Kontroll-Modul

Die restlichen drei Komponenten weisen einen eher unterstützenden Charakter auf. Sie haben folgende Aufgaben:

5. die Funktion des Kontroll-Moduls, einschließlich seiner Komponenten, zu überwachen und deren Abläufe zu steuern;
6. ein Modell des zu kontrollierenden Systems zu verwalten;
7. unterstützende Funktionen (z.B. Orbitberechnungen) den anderen Komponenten als Service anzubieten.

Je nach zu kontrollierendem System müssen verschiedene Kontroll-Module instanziert und mit Hilfe von Kontroll- und Datenflüssen verbunden werden. Die nachfolgende Abbildung 4-10 zeigt ein Beispiel eines fiktiven Satelliten zur Erdbeobachtung.

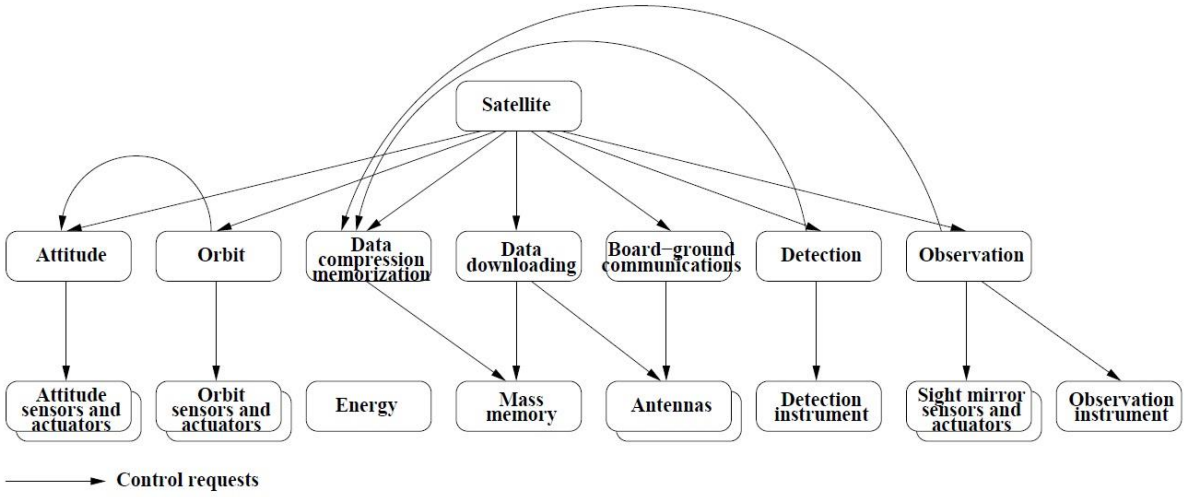


Abbildung 4-10: Steuerung eines Systems mit Kontroll-Modulen (Quelle: [44])

Jedes Kontroll-Modul ist einer der drei Hierarchiestufen (z.B. Satellit-Modul auf Stufe 1, Antennen-Modul auf Stufe 3) zugeordnet. Ein Modul darf andere Module nur dann per Kontrollfluss kontrollieren, wenn das zu kontrollierende Modul sich in der gleichen oder einer niedrigeren Hierarchiestufe wie das kontrollierende Modul befindet. Für Datenflüsse (nicht in der Abbildung dargestellt) gilt die Regel entgegen gesetzt.

In der einen Komponente des Kontroll-Moduls für die Entscheidungsfindung befinden sich die Funktionen, in denen das autonome Verhalten implementiert ist. Die Entwickler kamen zu dem Schluss, dass es unsinnig ist, wenn bei der Entscheidungsfindung Pläne für einen langen Zeitraum (auch Zeithorizont genannt) berechnet werden, da der zukünftige Zustand eines Satelliten nur sehr schwer modellierbar ist. Die Entscheidungsfindung in AGATA arbeitet daher nach dem in Abbildung 4-11 dargestellten Prinzip.

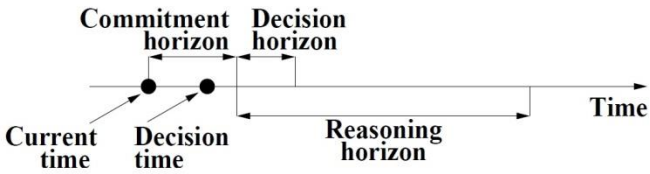


Abbildung 4-11: Zeitspannen des Planungsalgorithmus (Quelle: [44])

Es gibt drei Zeitbereiche. Der erste Bereich (Commitment) bezeichnet den Teil des Plans, der nicht mehr im Rahmen eines Planungsalgorithmus (Entscheidungsfindung) manipuliert werden darf. Das ist besonders dann sinnvoll, wenn die Aktionen in diesem Bereich bereits vom Bordcomputer des Satelliten initiiert wurden. Der zweite Bereich (Decision) kennzeichnet die Zeitspanne, in der stets ein Plan schnell vorliegen (bzw. berechnet werden) muss. Das gilt insbesondere dann, wenn der gesamte Plan aufgrund einer Zustandsänderung des Satelliten ungültig geworden ist. Der letzte Bereich (Reasoning) bezieht sich auf die Zeitspanne, für die ein optimierter Plan berechnet werden kann, sofern der zweite Bereich gültig ist. Während beim dritten Bereich die Optimierung im Vordergrund steht, ist beim zweiten Bereich die Antwortzeit bis zu seiner Gültigkeit von Bedeutung.

Den Entwicklern ist dabei aufgefallen, dass dieses Vorgehen dennoch ein paar Nachteile hat:

- Es kann nicht eindeutig bestimmt werden, wie groß die einzelnen Bereiche sein sollen.
- Dieses Verfahren reagiert zwar auf negative Einwirkungen, indem der zweite Bereich ungültig wird, aber es reagiert nicht auf positive Einwirkungen (z.B. erhöhte Verfügbarkeit von Ressourcen).

- Der zweite Bereich kann schnell an Gültigkeit verlieren, wenn der Zustand des Satelliten zu dynamisch ist.
- Es kann nicht garantiert werden, dass der zweite Bereich rechtzeitig berechnet werden kann (fehlende Echtzeitfähigkeit).

4.9 APSI

APSI (Advanced Planning and Scheduling Initiative) [46] ist ein vielseitiges, sich laufend fortentwickelndes, Framework zur Erstellung von Missionsplänen, nicht ausschließlich für Satelliten, sondern auch für andere Plattformen, wie zum Beispiel Roboter oder Teleskope. Es wurde im Rahmen einer Studie durch die ESA initiiert und anschließend umgesetzt.

Im Allgemeinen wurden bei diesem Ansatz zwei Ziele verfolgt:

- Softwareanwendungen zur Unterstützung der Missionsplanung sollen kostengünstiger und flexibler werden.
- Das Framework soll als Brückentechnologie dienen, um Lösungen aus dem Forschungsgebiet der künstlichen Intelligenz auf die Missionsplanung anwenden zu können.

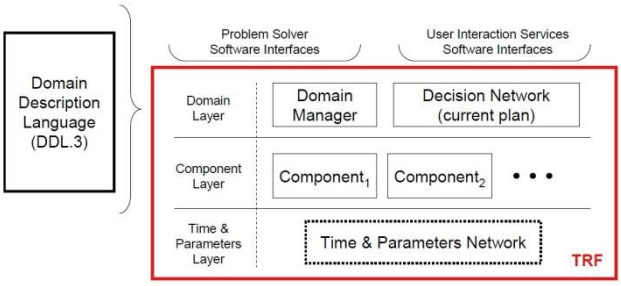


Abbildung 4-12: APSI Timeline Representation Framework (Quelle: [46])

Das Framework bietet einer übergeordneten Anwendung zur Missionsplanung Dienste an, indem es das allgemeine Verhalten eines zu planenden Systems als sogenanntes Domänenwissen modelliert, und für gewünschte Verhaltensabläufe (z.B. die Missionsplanung) eine Lösung berechnet, die bestimmte Vorgaben einhalten muss. Um dieser Aufgabe gerecht zu werden, wurde APSI mit einer dreischichtigen Architektur versehen (siehe Abbildung 4-12). Basisbausteine des Frameworks sind sogenannte generische Komponenten. Deren zeitliche Änderungen sind für die Lösung von besonderem Interesse. Daher verfügt jede Komponente über eine eigene Zeitlinie. Zustände einer Zustandsmaschine werden zum Beispiel auf diese Weise modelliert, dass zusätzlich weitere Informationen bzgl. der Verweildauer eines Zustandes oder die Voraussetzungen für einen Zustandsübergang in die entsprechende Komponente augmentiert werden. Auf diese Weise lassen sich funktionales und physikalisches Verhalten eines zu planenden Systems modellieren. Alle Komponenten sind in der mittleren Schicht der Architektur organisiert. Das in der obersten Schicht abgelegte Domänenwissen greift auf die untergeordneten Komponenten zu, um deren Wechselwirkungen untereinander zu beschreiben. Hierbei kommen verschiedene Softwaretechniken, wie Synchronisation oder Regeln, zum Einsatz. In der untersten Architekturebene befindet sich die eigentliche Funktionalität, um das modellierte Verhalten zu berechnen.

Wie bereits erwähnt wurde, reicht APSI alleine für die Aufgabe (z.B. Missionsplanung) nicht aus. In das Gesamtsystem müssen noch andere Teilnehmer eingebunden werden. Diese sind in der nachfolgenden Abbildung 4-13 schematisch dargestellt.

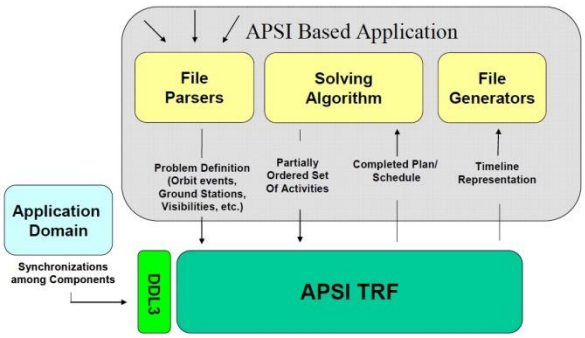


Abbildung 4-13: Aufbau eines kompletten Systems basierend auf APSI (Quelle: [46])

Das Framework muss mit dem notwendigen Domänenwissen (z.B. Verhalten des Satelliten) versorgt werden. Dieses wird formalisiert in einer speziellen Sprache (DDL, Domain Definition Language) verfasst und über eine entsprechende Schnittstelle dem Framework mitgeteilt. Genauso muss auf Anwendungsebene das eigentlich zu lösende Problem (z.B. die Randbedingungen eines Missionsplans) behandelt werden. Auch hierfür wurde eine spezielle Sprache (PDL, Problem Definition Language) definiert. Mit diesen Eingangsdaten kann das Framework versuchen, das Problem zu lösen. Als Ergebnis werden die oben erwähnten Zeitlinien der Komponenten ausgegeben. Nachfolgende Generatoren wandeln es in ein für die Anwendung lesbares Format um.

4.10 VAMOS

VAMOS (Verification of Autonomous Mission Planning On-Board a Spacecraft) [47] [48] [49] ist ein Experiment der FireBIRD Mission und wurde vom Deutschen Raumfahrtkontrollzentrum (GSOC) entwickelt. Sein Ziel besteht darin, einem der beiden an der Mission beteiligten Satelliten – BIROS – mit Hilfe einer Onboard-Software autonome Fähigkeiten zu verleihen.

BIROS ist ein Satellit zur Erdbeobachtung. Mit seiner Spektral- und Infrarotkamera ist er besonders dazu geeignet, heiße Flächen (Hotspots) auf der Erde, z.B. Waldbrände, zu detektieren.

Mit VAMOS soll der wissenschaftliche Gewinn (d.h. die Anzahl aussagekräftiger Bildaufnahmen von Hotspots) der Mission erhöht werden. Insbesondere sollen Gelegenheiten genutzt werden, wenn unerwarteter Weise Ressourcen im Satelliten zur Verfügung stehen. Diese Chancen ergeben sich unter anderem dadurch, dass der durch die Ablaufplanung (am Boden) berechnete Verlauf von Ressourcen vom tatsächlichen Verlauf abweichen kann.

Zur Optimierung bedient sich VAMOS zweier Mechanismen:

- OBoTiS (On-Board Timeline Selection)
- OBETTE (On-Board Event Triggered Timeline Extension)

Die wesentliche Funktionalität von OBoTiS besteht darin, dass er die aktuelle Kommandosequenz, auch *Base Timeline* genannt, mit Hilfe von Erweiterungen (*Timeline Extension*) ergänzt, wenn bestimmte Bedingungen erfüllt sind. Die nachfolgenden zwei Abbildungen (Abbildung 4-14 und Abbildung 4-15) verdeutlichen die Thematik.

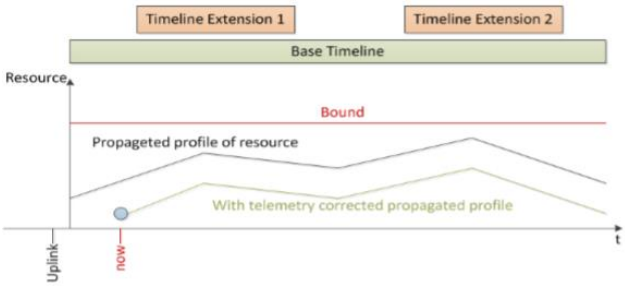


Abbildung 4-14: OBoTiS - Abweichung eines Ressourcenverlaufs, dadurch Erweiterung der Kommandosequenz möglich (Quelle: [47])

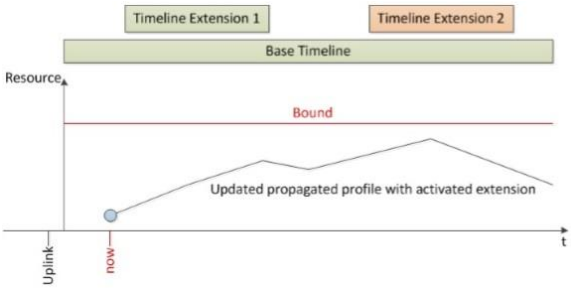


Abbildung 4-15: OBoTiS – Aktivierung einer Erweiterung der Kommandosequenz (Quelle: [47])

Der Fokus liegt auf der *Base Timeline* und dem damit modellierten zeitlichen Verlauf der beteiligten Ressourcen. Die Kommandosequenz wird bei jedem Kontakt mit der Bodenstation per Upload zum Satelliten aktualisiert. Der durch die Ablaufplanung berechnete Verlauf der Ressourcen bezieht sich jedoch auf einen bestimmten Zeitpunkt, der dadurch immer zwangsweise kurz vor dem Upload liegt. Der tatsächliche Verlauf kann im laufenden Betrieb abweichen. Daher kann es passieren, dass unerwarteter Weise Ressourcen zur Verfügung stehen. OBoTiS nutzt diese Gelegenheit, indem er beim Upload ebenfalls sogenannte *Timeline Extensions* zum Satelliten überträgt. Diese werden im Rahmen der Ablaufplanung am Boden berechnet und mit Bedingungen bzgl. der benötigten Ressourcen und der Zeiten zur Ausführung versehen. OBoTiS überprüft für jede Extension, ob kurz vor ihrer Ausführung die benötigten Ressourcen zur Verfügung stehen, und aktiviert diese gegebenenfalls. Zwar werden bei VAMOS die *Base Timeline* und die *Timeline Extensions* weiterhin am Boden berechnet, allerdings findet im laufenden Betrieb eine autonome, an die gegebene Situation des Satelliten angepasste, Manipulation der Kommandosequenzen statt.

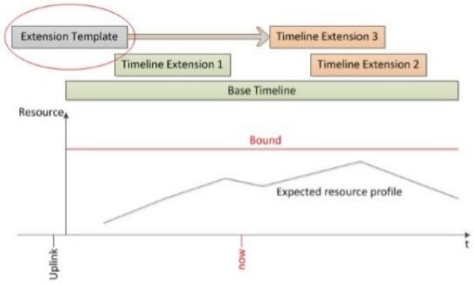


Abbildung 4-16: OBETTE – Erstellung einer Erweiterung (Quelle: [47])

Die Erzeugung von *Timeline Extensions* ist jedoch nicht nur auf den Boden beschränkt. Mit OBETTE lassen sich neue Erweiterungen onboard erstellen (siehe Abbildung 4-16). Die Vorgehensweise ist ereignisorientiert. Die Auslösung eines Ereignisses erfolgt durch Komponenten/Mechanismen außerhalb von VAMOS, im Fall von BIROS zum Beispiel durch das Verwerfen einer Bildaufnahme auf Grund einer zu starken Wolkenbedeckung. Daraus folgend soll eine neue *Timeline Extension* erzeugt werden, um mit dem Satelliten eine neue Bildaufnahme durchzuführen. OBETTE bedient sich dazu vordefinierter Vorlagen und setzt entsprechende Parameter (z.B. Ausführungszeiten) ein. Bei der Erstellung muss ebenfalls der Verlauf der Ressourcen berücksichtigt werden.

4.11 NEAT

NEAT (Near-optimal Evolutionary Autonomous Task-manager) [50], [51] ist ein Planungssystem, das einen genetischen Ansatz verfolgt. Es wurde vom Surrey Space Center der Universität von Surrey (Guildford, England) entwickelt. Zwar wurde das System bisher noch nicht an einem realen Einsatz im

Orbit verifiziert, jedoch wurden einige Falluntersuchungen mit echten Telemetrie-Daten des UK-DMC (Desaster Monitoring Constellation) Satelliten erfolgreich durchgeführt.

Der Aufbau und die Funktionsweise des Planungssystems wird anhand der folgenden Abbildungen (Abbildung 4-17 und Abbildung 4-18) erläutert.

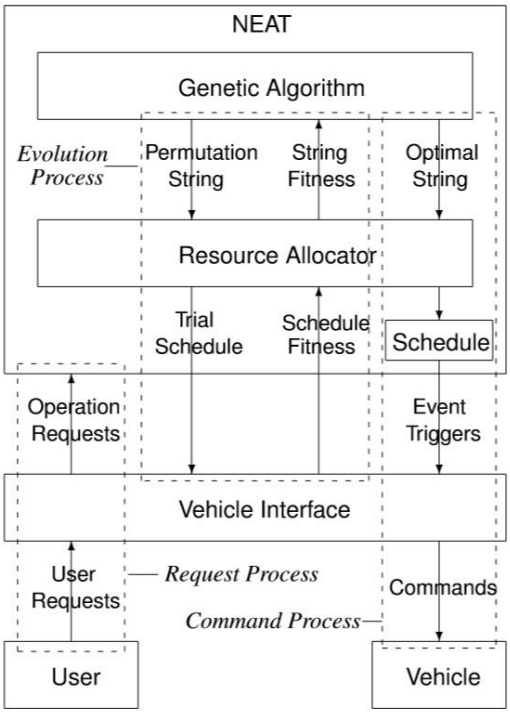


Abbildung 4-17: NEAT – Aufbau (Quelle: [50])

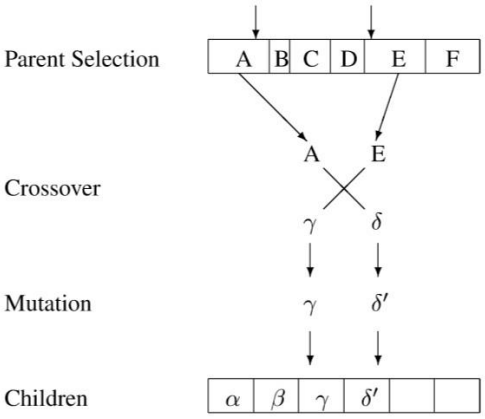


Abbildung 4-18: Abbildung 4-19: NEAT – genetischer Algorithmus (Quelle: [50])

Das übergeordnete Ziel einer mit NEAT ausgestatteten Satellitenmission besteht darin, Anfragen des Operators entgegenzunehmen und diese in die autonome Planung des Satelliten einzubeziehen. Das Ergebnis ist eine Sequenz von eigenständig anbord erstellten Kommandos. Die Einbindung von NEAT in das Gesamtsystem erfolgt über eine spezifische Schnittstelle (*Vehicle Interface*). Diese hat die Aufgaben, die Anfragen des Operators in Operationen - ein für NEAT konformes Format (*Operation Requests*) - zu übersetzen und die Ausgabe in Form von sogenannten Ereignissen in die eigentlichen Kommandos für den Satelliten zu übersetzen. Ereignisse haben einen direkten Bezug zu Operationen. Zum Beispiel kann ein Ereignis den Beginn einer Operation einleiten.

NEAT verwaltet intern einen Gen-Pool bestehend aus einer Menge von Gen-Strängen, deren einzelne Abschnitte als Gene bezeichnet werden, und jeweils einer Operation entsprechen.

NEAT arbeitet iterativ, d.h. es versucht, in Zyklen den aktuellen Plan zu verbessern. Insgesamt besteht ein Zyklus aus vier Schritten:

1. Der erste Schritt eines Zyklus besteht darin, sämtliche dem Zyklus zugewiesenen Ereignisse zur weiteren Verarbeitung auszulösen. Ereignisse, die den Beginn einer Operation einleiten, haben zur Folge, dass das entsprechende Gen aus dem Genpool bzw. den Gen-Strängen entfernt wird. Ab diesem Zeitpunkt ist die betroffene Operation von der Optimierung durch den Algorithmus ausgeschlossen.
2. Im zweiten Schritt werden die an NEAT gestellten Anfragen seitens des Operators ausgewertet. In der Regel werden dabei neue Operationen in die Planung aufgenommen. Hierbei wird für jede neue Operation ein Gen erstellt und in jeden Gen-Strang im Gen-Pool eingefügt. Die Position ist zufällig.
3. Im dritten Schritt wird die Population des Gen-Pools evolutionär verändert (siehe Abbildung 4-18). Dabei werden jeweils zwei Stränge gekreuzt und anschließend einzeln mutiert. Beide

Vorgänge finden mit einer voreingestellten Wahrscheinlichkeit statt. Dieser Schritt wird mehrmals wiederholt, sodass mehrere Generationen des Gen-Pools erstellt werden.

4. Im letzten Schritt wird aus dem Gen-Pool der Gen-Strang mit der besten Bewertung selektiert, und von diesem ein neuer Plan generiert.

Bei der Planerstellung müssen auch die für die Operationen zur Verfügung stehenden Ressourcen berücksichtigt werden. Der evolutionäre Ansatz liefert zunächst für die Operationen nur eine zeitliche Reihenfolge, die im Rahmen der nachträglichen Allokation verändert werden kann. Hierzu bedient sich der Algorithmus der Dienste eines Ressourcen-Zuteilers (*Resource Allocator*). Dieser muss für einen vorgegebenen Plan zum einen den Verlauf der Ressourcen propagieren und zum anderen eine Bewertung vornehmen. Das Ergebnis hiervon stellt die Entscheidungsgrundlage (4. Schritt) für den evolutionären Algorithmus dar.

Die Propagation der Ressourcen basiert auf Modellen. Die realen Verläufe der Ressource können daher von den berechneten Verläufen abweichen. Um diesen Abweichungen entgegenzuwirken und sie so gut wie möglich auszugleichen, verwendet NEAT ein neuronales Netzwerk (siehe Abbildung 4-20).

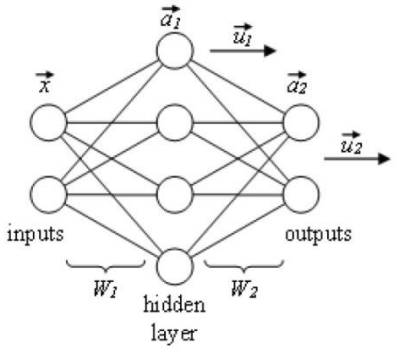


Abbildung 4-20: NEAT – Neuronales Netzwerk zur Korrektur der Res.-Propagation (Quelle: [51])

Für jede modellierte Ressource weist das neuronale Netzwerk einen Eingang und einen Ausgang auf. Am Eingang wird der berechnete Wert der jeweiligen Ressource angelegt. Der Ausgang liefert einen Korrekturwert, der der oben genannten Abweichung entgegenwirkt. Vor dem Einsatz muss das Netzwerk trainiert und gewichtet werden.

4.12 Auswertung und Abgrenzung

Im letzten Abschnitt des Kapitels soll eine Analyse bzgl. des aktuellen Stands der Technik und der Forschung aufgezeigt werden. Folgende Fragen stehen dabei im Vordergrund:

- Wurde die Zielsetzung (siehe Abschnitt 1.6) bereits vollständig oder auch nur teilweise erreicht?
- Können Beziehungen und Abgrenzungen zu bereits vorhandenen Erkenntnissen gezogen werden?
- Warum beinhaltet der aktuelle Stand der Forschung und der Technik keine Lösung für das vorliegende Problem (siehe Abschnitt 1.3)?
- Was ist an dem Forschungsziel neu, bzw. worin unterscheidet es sich von bekannten Problemlösungen?

Mit jedem der in den Abschnitten 4.2 bis 4.11 vorgestellten Planungssystemen lässt sich autonomes Verhalten realisieren. In diesem Sinne ist die Zielsetzung bereits zum Teil durch den Stand der Technik und der Forschung erreicht.

Eine genauere Untersuchung zeigt jedoch auf, dass die Zielsetzung nicht vollständig erreicht wurde. Insbesondere sollen folgende Eigenschaften von autonomen Planungssystemen gegenübergestellt werden:

- Entscheidungsfindung
  Die Fähigkeit, Entscheidungen zu treffen.
- Ablaufplanung
  Die Fähigkeit, Entscheidungen mit vorhandenen Mitteln (z.B. Ressourcen) und Gegebenheiten (z.B. Beschränkungen) in einen Plan umzusetzen.
- Ausführung
  Die Fähigkeit, einen Plan in Kommandos umzusetzen und mit der Bordsteuerung des Satelliten abzustimmen.
- Flexibilität (Mission)
  Eine Angabe, ob das Planungssystem an verschiedene Satellitenmissionen angepasst werden kann.
- Anpassbarkeit (Kommandierung)
  Die Fähigkeit, die Verhaltensweise des Planungssystems per Kommandierung zu ändern.
- Kleinstsatellit
  Die Eignung des Planungssystems für Kleinstsatelliten (<20 Kg)

	Entschei-dungsfindung	Ablaufplanung	Ausführung	Flexibilität (Mission)	Anpassbarkeit (Kommandierung)	Kleinst-satellit
ASPEN / CASPER	Ja	Ja	Nein	Ja	Nein	Nein
ASE	Ja	Ja	Ja	Ja	Nein	Nein
Reaktive/ Deliberative	Nein	Ja	Ja	Ja	Nein	Nein
ADEPT	Ja	Ja	Ja	(Ja)	Nein	Ja
AGATA	Ja	Ja	Ja		Nein	-
APSI	Ja	Ja	Ja	Ja (Boden)	-	Nein
VAMOS	Ja	Ja (Boden)	Ja	Nein	Ja	Nein
NEAT	Nein	Ja	Nein	-	-	Nein

Tabelle 4-2: Gegenüberstellung autonomer Planungssysteme

Wie in der Zielsetzung (Abschnitt 1.6) erläutert wurde, erhebt die Dissertation den Anspruch, das autonome Planungsproblem zu lösen. Dieses wurde zwar im Kern, d.h. bei der Entscheidungsfindung, der Ablaufplanung und der Planausführung, bereits prinzipiell gelöst, hingegen weist jedoch die Anwendung der Systeme in einer Satellitenmission noch einige gravierende Unzulänglichkeiten auf. Davon sind insbesondere die (fehlende) Flexibilität und die (mangelhafte) Anpassbarkeit betroffen. Als Folge hiervon müssen laufende Änderungen der Missionsplanung (z.B. neue Vorschriften für die Entscheidungsfindung) in der Regel über riskante Softwareupdates durchgeführt werden.

Der Grund für die Unzulänglichkeiten (Flexibilität und Anpassbarkeit) besteht aus Sicht des Autors darin, dass autonome Planungssysteme eher als „Beiprodukte“ von heutigen Satellitenmissionen entworfen werden. Sie sind daher im höchsten Maße auf ihre jeweiligen Satellitenmissionen optimiert. Tatsächlich werden autonome Planungssysteme derzeit in der Regel am Ende einer Mission, wenn der Totalausfall eines Satelliten in Kauf genommen werden kann, erprobt. Es kommt selten vor, dass sie von Grund auf, losgelöst von speziellen Satellitenmissionen, auf Flexibilität und Anpassbarkeit konzipiert werden. Stattdessen werden sie für eine Mission grundlegend entworfen, und allenfalls für weitere Missionen angepasst. Strukturelle Beschränkungen in der Architektur von Planungssystemen sind die Folge.

Des Weiteren sind die vorgestellten Planungssysteme, mit Ausnahme von ADEPT, nicht für den Einsatz in Kleinstsatelliten vorgesehen. Der von den Systemen geforderte Bedarf an elektrischer Leistung, Energie, Masse und Volumen – gegeben durch die Software und der zugrunde liegenden Hardware (Plattform) – übersteigt die zur Verfügung stehenden Ressourcen heutiger Kleinstsatelliten. Der Grund hierfür liegt darin, dass die heute existierenden Planungssysteme im Rahmen größerer Satellitenmissionen entworfen wurden, und daher nie den Beschränkungen von Kleinstsatelliten unterworfen wurden. Dieser Umstand kann sogar dazu führen, dass bereits heute verfügbare Lösungen für das Planungsproblem hinsichtlich der Entscheidungsfindung, Ablaufplanung und Planausführung unbrauchbar sind, da für die verwendeten Algorithmen auf einem Kleinstsatelliten schlicht die benötigten Ressourcen (z.B. Rechenleistung und Speicher) fehlen könnten.

Zusammengefasst soll sich das zu entwickelnde Planungssystem vom aktuellen Stand der Technik und der Forschung dahingehend abheben, als dass es sowohl die eigentlichen autonomen Planungen durchführen kann, als auch zusätzlich wichtige Aspekte hinsichtlich der Anwendung im laufenden Betrieb, wie zum Beispiel die Flexibilität und die Anpassbarkeit, vereinigt. Das Planungssystem soll des Weiteren den Anforderungen (Leistung, Energie, Masse, Volumen) von Kleinstsatelliten (min. Nanosatelliten) genügen.

5 Methode

Zur Lösung der im Abschnitt 1.3 beschriebenen Problemstellung müssen eine oder mehrere geeignete wissenschaftliche Methoden ausgewählt und angewendet werden. Da das Problem technischer Natur ist und im Kontext eines Satellitensystems auftritt, ist die Lösung selbst, wie im Abschnitt 1.4 angedeutet wurde, ein technisches System, und bedarf daher einer Systementwurfsmethode. Diese soll in ihrer Anwendung zu Systemen führen, die das autonome Planungsproblem lösen.

Die verwendete Systementwurfsmethode basiert auf der MBSE-Methode (engl. model based system engineering). Im Zentrum dieser Technik steht das Modell. Es ist eine Abstrahierung eines echten Systems, das es zu entwickeln gilt. Der Hauptgedanke besteht darin, alle Informationen und Dokumente (z.B. Spezifikationen, Implementierungen, Dokumentationen) von diesem Modell abzuleiten, vorausgesetzt, es besitzt den nötigen Detailgrad. MBSE hat den Vorteil, dass Änderungen im Modell direkt auf die generierten Dokumente reflektiert werden können. Dadurch wird eine Konsistenz aller Dokumente zueinander gewährleistet. Es sei anzumerken, dass in der vorliegenden Arbeit der Automatismus nicht vollständig umgesetzt wurde. Zum Beispiel wurde der Quellcode größtenteils manuell erstellt. Dennoch ist die MBSE-Methode sehr hilfreich, da sie noch weitere Aspekte für den Systementwurf abdeckt:

1. Modellierungssprache
2. Modellierungsmethode
3. Modellierungswerkzeug

Modellierungssprachen bieten die Möglichkeit, Modelle formal zu beschreiben. Mit ihnen können in der Regel das dynamische Verhalten und die statische Struktur eines Systems modelliert werden. Es existieren verschiedene Modellierungssprachen, beginnend mit der bekannten UML (Unified Modelling Language), gefolgt von der SysML (System Modelling Language), Petri-Netzen, SOMF (Service Oriented Modelling Framework), ADL (Architecture Description Language), SA/SD (Structured Analysis / Structured Design), usw. Die Modelle der zu entwerfenden Systeme werden in der SysML, eine Erweiterung der UML (durch die Verwendung eines UML-Profils), erstellt. Die UML ist lediglich auf die Beschreibung von Software ausgelegt, während die SysML zusätzlich auch andere Aspekte (z.B. Hardware) unterstützt. Für die Modellierung der Systeme werden verschiedene Typen von SysML-Diagrammen, die im Abschnitt 2.4 vorgestellt wurden, verwendet.

Ein Modell besteht aus mehreren Elementen, die entweder den statischen oder dynamischen Aufbau des zu entwickelnden Systems beschreiben. Dazu gehören unter anderem Anforderungen, Blöcke, Teile, Zustandsmaschinen, Zustände, Aktivitäten, Aktionen, Assoziationen, Objekte, Instanzen, Ereignisse, Operationen, Nachrichten, usw.

Es sei noch anzumerken, dass ein einzelnes Diagramm lediglich eine Sicht auf das Modell darstellt und nicht notwendigerweise jedes Detail abbildet. Dies gilt besonders für die Diagramme der nachfolgenden Kapitel (6.Konzept, 7.Realisierung).

Die Modellierungsmethode beschreibt die Vorgehensweise, wie das Modell im Laufe des Entwurfs aufgebaut werden soll. Es existieren verschiedene Arten von Methoden, die aber in der Regel von einer der drei folgenden Entwurfsmethoden aus der Software- und Systementwicklung abstammen:

- Wasserfall-Methode
- Spiral-Methode
- V-Methode

Für die zu entwickelnden Systeme wird eine Kombination des klassischen Spacecraft System Design und der V-Methode als Systementwurfsmethode angewendet. Sie ist in der nachfolgenden Abbildung 5-1 dargestellt und spiegelt sich in der Kapitelstruktur des Abschnitts 7 (Realisierung) wieder.

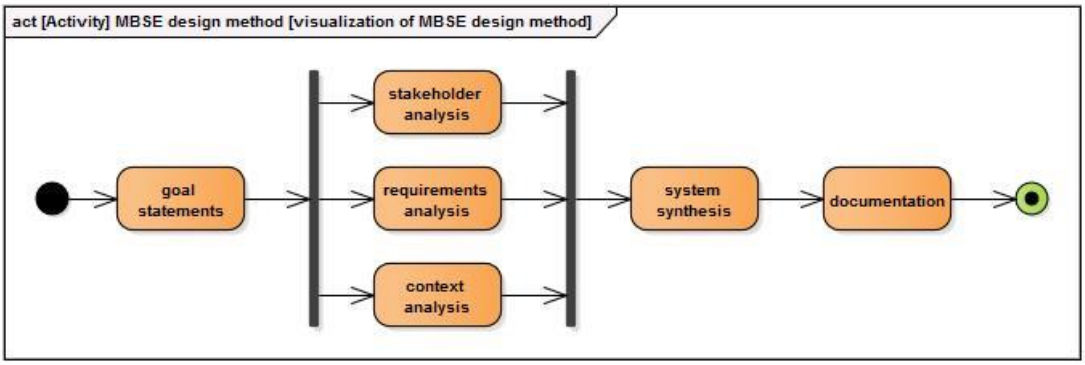


Abbildung 5-1: Prozessschritte der verwendeten Entwurfsmethode

Der Entwurf beginnt immer mit einer Beschreibung der Ziele (*goal statements*). Meistens handelt es sich um einen in menschlicher Sprache verfassten Text, in dem die wichtigsten Ziele und Anforderungen an das zu entwickelnde System festgehalten werden. Alle im Abschnitt 7 (Realisierung) vorgestellten Systementwürfe beginnen daher mit einer einleitenden Zielstellung. Als nächstes müssen drei verschiedene Arten von Analysen, die auch gleichzeitig ablaufen können, durchgeführt werden:

- Analyse der Stakeholder (*stakeholder analysis*)
  Das zu entwickelnde System wird in ein bestimmtes Umfeld eingesetzt, und muss mit Teilnehmern (Stakeholder) interagieren. Bei den Stakeholdern kann es sich um andere technische Systeme, Menschen, Normen, Institutionen, Gruppen, usw., handeln. Ziel der Analyse besteht darin, sämtliche relevanten Stakeholder des zu entwickelnden Systems zu identifizieren.
- Analyse des Kontexts (*context analysis*)
  Das zuvor erwähnte Umfeld wird auch Kontext genannt. Bei der Analyse des Kontexts geht es vor allem darum, Schnittstellen zwischen den Stakeholdern und dem zu entwickelnden System zu identifizieren. Dabei sollen auch die Interaktionen, die über diese Schnittstellen stattfinden, erfasst werden. Der Kontext ist in der Regel ein SoS (System of Systems). Daher kann es durchaus sein, dass an einer Interaktion mehrere Stakeholder beteiligt sind.
- Analyse der Anforderungen (*requirements analysis*)
  Jeder Stakeholder kann Anforderungen an das zu entwickelnde System stellen. Diese müssen formal erfasst werden. Bei der Entwurfsmethode wird in Anlehnung an [52] zwischen vier verschiedenen Arten von Anforderungen unterschieden:

  - Nutzeranforderungen (User Requirement)
  - Funktionale Anforderungen (Functional Requirement)
  - Qualitätsanforderung (Performance Requirement)
  - Randbedingungen (Design Constraint)

  Nutzeranforderungen bilden den Ausgangspunkt der Analyse. Sie beschreiben die Anforderungen aus Sicht der eigentlichen (End-)Nutzer und stellen eine Abstrahierung dar, indem sie Details auf Systemebene ausblenden. Auf der darunter liegenden Abstraktionsebene befinden sich die restlichen Anforderungen, angefangen mit den funktionalen Anforderungen. Diese beziehen sich auf eine Funktionalität (z.B. eine Rechenoperation), die das zu entwickelnde System bereitstellen muss. Sie lassen sich durch Testfälle validieren. Qualitätsanforderungen hingegen sind als Ergänzung zu verstehen. Sie dienen dazu, funktionale Anforderungen zusätzlich einzuschränken (z.B. Dauer der Rechenoperation < 1 Sekunde). Dadurch erreichen bereitgestellte Funktionalitäten eine gewünschte Qualität. Randbedingungen haben ebenfalls einschränkenden Charakter und können sich sowohl auf funktionale als auch auf qualitative Anforderungen beziehen. Sie sind

unabhängig vom zu entwickelnden System und beschreiben Gegebenheiten (z.B. Infrastruktur, Kostengrenze).
Im Allgemeinen ist es bei der Erstellung von Systemen üblich, den Grad der Verbindlichkeit von Anforderungen durch Modalverben zu spezifizieren. In der vorliegenden Arbeit wird zwischen den drei folgenden Varianten unterschieden:

- muss: Es gilt, die Anforderung vollständig ohne Abweichungen in der gewünschten Qualität umzusetzen.
- soll: Es gilt, die Anforderung nach Möglichkeit vollständig umzusetzen. Abweichungen von der gewünschten Qualität sind jedoch zulässig.
- sollte: Es ist wünschenswert, die Anforderung zu erfüllen. Das System würde dadurch leistungsfähiger werden. Eine Umsetzung ist jedoch nicht notwendig.

Das Ergebnis der Analyse ist eine Spezifikation, in der alle Anforderungen formal aufgeführt sind. Der nächste Schritt des Entwurfsprozesses besteht darin, die Spezifikationen als Input im Rahmen der nachfolgenden Entwurfssynthese zu verarbeiten.

Die Synthese hat die Aufgabe, einen zur Spezifikation konformen Block (SysML-Element für ein System) zu erstellen, der sowohl das gewünschte dynamische Verhalten (Dienste und Funktionalitäten) als auch die erforderliche statische Struktur (Teilblöcke und deren Verschaltung untereinander, Schnittstellen) aufweist. Die Synthese arbeitet rekursiv und startet eine Rekursion, sobald bei der Strukturierung des Blockes Teilblöcke als hierarchische Kindelemente notwendig werden, diese aber noch nicht definiert bzw. entworfen wurden. Die Arbeitsweise verläuft nach dem Top-Down-Prinzip, d.h. das zu entwerfende System wird zuerst in seiner Gesamtheit als Block betrachtet und dann in seine Teilblöcke strukturell aufgegliedert. Die nachfolgende Abbildung 5-2 verdeutlicht die einzelnen Schritte der Synthese.

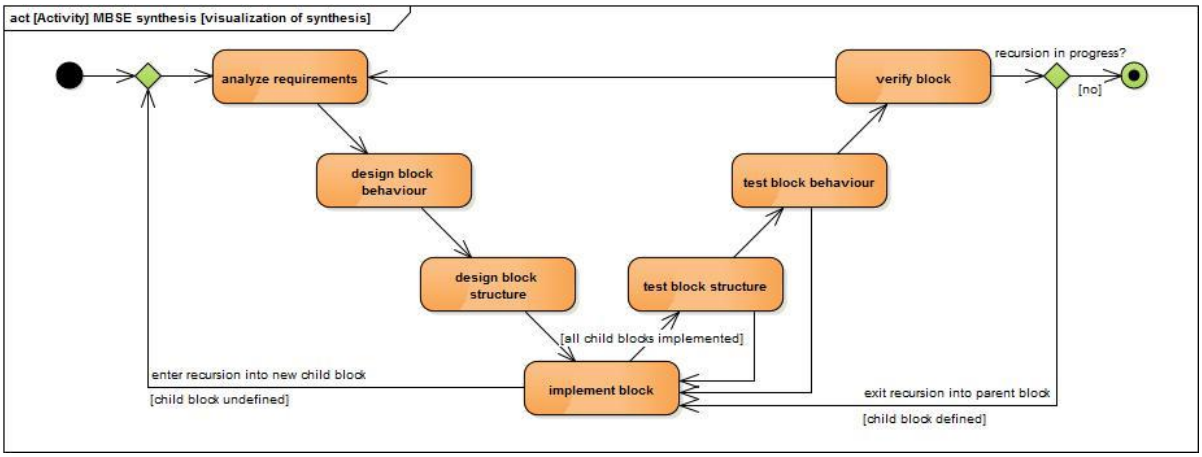


Abbildung 5-2: Prozessschritte während der Entwurfssynthese

Jeder Durchlauf, einschließlich aller Rekursionen, startet mit einer Analyse der Anforderungen an den zu synthetisierenden Block. Der erste Durchlauf beginnt daher mit einer Untersuchung der Spezifikation auf Systemebene. Jede Anforderung muss qualitativ und quantitativ genau beschrieben worden sein, damit sie später im Rahmen von Blocktests verifiziert werden kann. Nach erfolgreicher Analyse wird das dynamische Verhalten des Blocks unter anderem mit Hilfe von Anwendungsfällen, Aktivitäten, Aktionen, Zuständen und Sequenzen modelliert. Falls die Funktionalität des zu entwerfenden Blocks zu komplex ist, kann sie im Rahmen der nachfolgenden Blockstrukturierung auf mehrere Teile aufgeteilt werden. Jedes Teil wird dabei wieder durch einen Block definiert. Aufgabe des Strukturentwurfs ist es, sicherzustellen, dass jeder Aspekt der Blockfunktionalität durch den Block selber, durch einen seiner Teile oder durch eine oder mehrere Interaktionen zwischen den Teilen übernommen wird. Die nachfolgende Implementierung realisiert die Struktur- und

Verhaltensdefinitionen des Blockes. Wie oben bereits erwähnt wurde, kann der Fall eintreten, dass für einen Block, der einen (Struktur)Teil eines übergeordneten Blockes definiert, keine Blockdefinition (also weder Verhaltens- noch Strukturbeschreibung) vorhanden ist. In diesem Fall wird eine Rekursion, beginnend mit der Analyse der Anforderungen an diesen Block, ausgelöst. Bei hinreichend komplexen Systemen können mehrere geschachtelte Rekursionen auftreten. Sobald die Rekursion abgeschlossen ist, und damit alle verwendeten Blöcke definiert sind, kann die eigentliche Implementierung stattfinden. Die dafür verwendeten Techniken variieren je nach Art des Blockes:

- Blöcke, die eine Software modellieren, werden in einer Programmiersprache codiert.
- Blöcke, die eine Hardwareschaltung modellieren, werden schematisiert, gelayoutet und geroutet.
- Blöcke, die eine RTL (Register-Transfer-Logik, z.B. bei FPGAs) modellieren, werden mit Hilfe spezieller Entwicklungsumgebungen umgesetzt.
- Blöcke, die eine Mechanik modellieren, werden konzipiert und gefertigt.

Im Anschluss an die Implementierung werden die Blöcke getestet. Jeder entwickelnden Entwurfsphase ist ein entsprechender Test zugeordnet. Der Strukturtest hat zum Ziel, sowohl die Instanziierung der Teile im Block als auch deren statische Verknüpfungen untereinander zu überprüfen. Im Anschluss findet der Verhaltenstest statt. Mit diesem Test wird die Funktionalität des Blocks untersucht. Dieser Vorgang schließt die untereinander stattfindenden Interaktionen der im Block befindlichen Teile mit ein. Die Test-Prozedur endet mit einer formalen Verifikation der Anforderungen an den Block. Zu diesem Zweck werden spezielle Testfälle erstellt. Sie sollen den formalen Beweis liefern, dass der Block gemäß der Spezifikation einwandfrei funktioniert. Nach erfolgreicher Testphase terminiert die Entwurfsmethode den aktiven Rekursionsdurchlauf und fährt mit der Implementierung des Elternblocks fort. Die gesamte Synthese endet, sobald es sich bei dem erfolgreich getesteten Block um das Gesamtsystem (d.h. keine aktiven Rekursionsdurchläufe, Top-Ebene) handelt.

Im Anschluss an die Synthese wird in einem letzten Schritt der Entwurfsmethode die Dokumentation zu dem entworfenen System erstellt. Mit Hilfe eines geeigneten Modellierungstools lassen sich Teile der Dokumentation, insbesondere die Beschreibung der erzeugten Elemente im Modell sowie sämtliche Diagramme, automatisch generieren.

Als Modellierungswerkzeug wurde die Software Enterprise Architekt (Hochschullizenz) von Sparks Systems verwendet. Sie unterstützt unter anderem die Verwendung von sowohl der SysML 1.3 als auch der UML 2.4.

6 Konzept

Dieser Abschnitt stellt ein Konzept vor, wie das in Abschnitt 1.3 eingeführte autonome Planungsproblem gelöst werden kann und erläutert dabei, wodurch sich dieser Ansatz vom aktuellen Stand der Technik abhebt und eine wissenschaftliche Verbesserung darstellt.

Ausgangspunkt für den Entwurfsprozess des zu entwickelnden autonomen Planungssystems (APS) ist eine Liste von Anforderungen, die es erfüllen muss:

1. Das autonome Planungssystem (APS) muss dem Bordcomputer (OBC) des Satelliten autonome Funktionalitäten als Dienst anbieten. Die im Abschnitt 1.4 beschriebene Vision stellt die Grundlage hierfür dar. Diese Dienste können vom OBC genutzt werden, um autonomes Verhalten zu realisieren.
2. Das autonome Planungssystem muss eine Schnittstelle aufweisen, über der es kommandiert, konfiguriert und überwacht werden kann.
3. Das autonome Planungssystem muss in der Lage sein, das Verhalten des Satelliten über Manipulationen der Kommandoliste im OBC verändern zu können.
4. Das autonome Planungssystem muss ein zielorientiertes Planungsverhalten unterstützen.
5. Das autonome Planungssystem muss ein ereignisorientiertes Planungsverhalten unterstützen.
6. Das autonome Planungssystem muss die Möglichkeit bieten, Beschränkungen zu unterstützen.
7. Das autonome Planungssystem muss sehr schnell auf Ereignisse reagieren können.

Als nächstes sollen die benötigten Funktionalitäten (siehe Abbildung 6-1) aus der Sichtweise des Operators dargestellt werden.

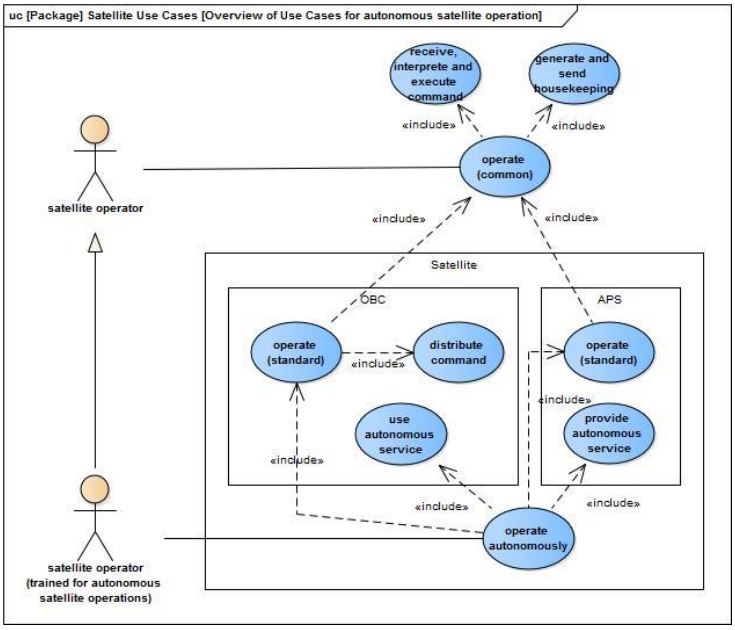


Abbildung 6-1: Anwendungsfälle für den autonomen Betrieb eines Satelliten

Der autonome Betrieb (*operate autonomously*) stellt eine Erweiterung des konventionellen Betriebs (*operate standard* und *operate common*) dar. Zu diesem gehören unter anderem die Verarbeitung von Kommandos und die Bereitstellung der Telemetrie. Die eigentliche Ergänzung besteht in der Anbindung des autonomen Dienstes. Zum einen muss das autonome Planungssystem (APS) dazu solch einen Dienst bereitstellen (*provide autonomous service*), zum anderen muss der Board-Computer (OBC) jenen Dienst auch tatsächlich nutzen (*use autonomous service*).

Unter Berücksichtigung der Information im Kapitel 5 führt ein Synthetisieren des in den Anwendungsfällen beschriebenen Verhaltens des Satelliten zu einer der in der nachfolgenden Abbildung 6-2 dargestellten Struktur.

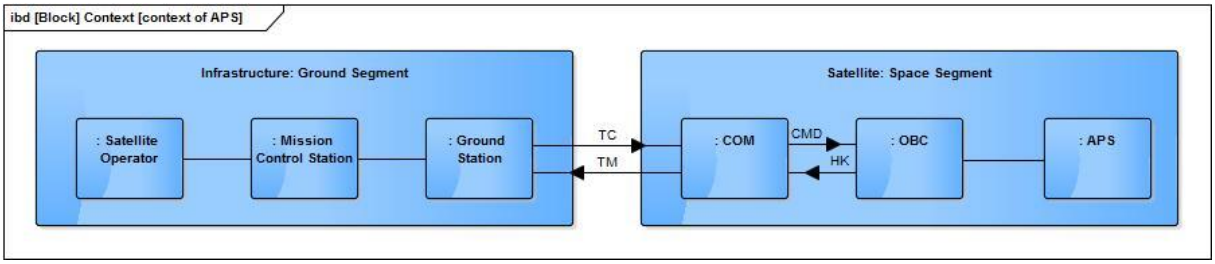


Abbildung 6-2: In die Autonomie involvierte Teilnehmer/Komponenten

Wie bereits im Abschnitt 2.1 erläutert wurde, sind beim Prozessieren der Kommandos und der Telemetrie für den Satelliten mehrere Teilnehmer involviert. Der Austausch der Telekommandos (TC) und der Telemetrie (TM) erfolgt über die Bodenstation. Gesteuert wird der Datenverkehr von Operatoren aus einem Kontrollzentrum heraus. Die Telekommandos werden vom COM-Modul empfangen, dekodiert und nach einer Verifikation an den Bordcomputer zur Ausführung weitergereicht. Dies wird als konventionelle Kommandierung bezeichnet. Hierbei ist ersichtlich, dass das Bodensegment die Quelle (d.h. den Satellitenoperator) für die Kommandos, und das Raumsegment die Senke enthält. In entgegen gesetzter Richtung sendet der Bordcomputer Telemetrie. Die bisher erläuterte Infrastruktur reicht aus, um die Anwendungsfälle zur Kommandierung und Telemetrierung abzudecken. Für die übrigen Anwendungsfälle, die die eigentlichen autonomen Funktionalitäten beherbergen, muss, wie in den Anforderungen gefordert, eine neue Komponente eingeführt werden. Es handelt sich hierbei um das autonome Planungssystem (APS). Der Grund dafür liegt darin, dass beim regulären autonomen Betrieb des Satelliten eine Kommandierung über die logische Kette, beginnend beim Operator bis hin zum Bordcomputer, nicht möglich ist. Es ist daher eine logische Konsequenz, dass die Kommandos intern im Satelliten generiert werden müssen. Diese Aufgabe wird durch das APS wahrgenommen. Dieser Vorgang wird auch als autonome Kommandierung bezeichnet. Das APS wird direkt vom OBC kontrolliert, und bietet diesem seine Dienste an. Im Autonomiebetrieb ersetzt das APS den Satellitenoperator als Quelle für die Kommandos. Die genauen Interaktionsmöglichkeiten müssen jedoch erst in einem weiteren Entwurfsschritt spezifiziert werden. Dazu werden als nächstes die vom OBC geforderten Funktionalitäten per Anwendungsfälle festgehalten. Eine graphische Repräsentation ist in der nachfolgenden Abbildung 6-3 dargestellt.

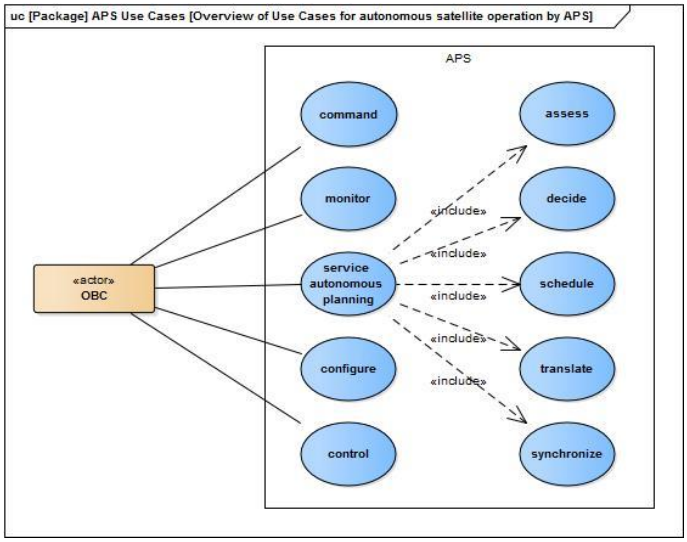


Abbildung 6-3: Anwendungsfälle des APS-Moduls

Damit der Satellit autonom operieren kann, muss das APS dem OBC folgende Funktionalitäten anbieten:

- Kommandierung: Der OBC sendet Kommandos, die vom APS verifiziert, decodiert, interpretiert und ausgeführt werden.
- Überwachung: Das APS sendet dem OBC nach zuvor konfigurierten zeitlichen Abständen Informationen (Telemetrie) über seinen internen Zustand.
- Konfiguration: Bestimmte Parameter des APS können wahlweise eingestellt werden. Dadurch lässt sich in gewissem Maße das Verhalten des APS anpassen.
- Steuerung: Das APS besitzt eine Steuerung, die vom OBC funktionell angestoßen wird.
- Autonomes Planen: Im Rahmen des autonomen Planungsdienstes muss das APS weitere Teilfunktionalitäten realisieren
    - Die autonome Planung muss die aktuelle Lage des Satelliten einschätzen können.
    - Die autonome Planung muss Entscheidungen treffen können, welche Aktivitäten der Satellit ausführen soll. Die Entscheidungsfindung kann sowohl zielorientiert als auch ereignisorientiert ausgerichtet sein.
    - Die Aktivitäten müssen in ihren Abläufen geplant werden.
    - Die geplanten Aktivitäten müssen in Sequenzen von Kommandos übersetzt werden.
    - Die übersetzten Kommandos müssen mit dem Inhalt der Kommandoliste vom OBC synchronisiert werden.

Die in den vorherigen Anwendungsfällen deklarierten Funktionalitäten werden in einem anschließenden Entwurfsschritt als Aktivitäten definiert. Die nachfolgende Abbildung 6-4 zeigt das Ergebnis in Form eines SysML-Aktivitätendiagramm.

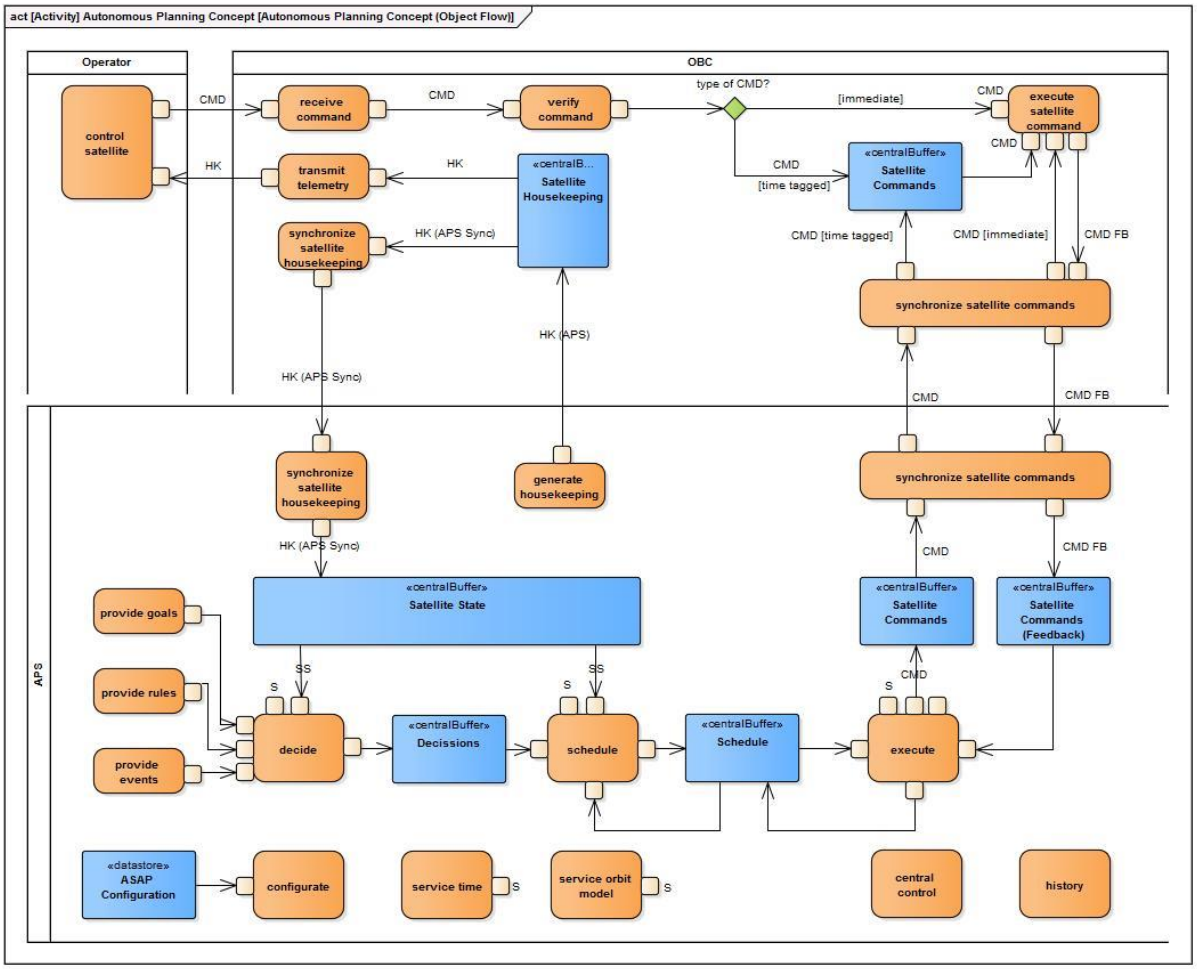


Abbildung 6-4: Darstellung der Interaktion zum autonomen Planen als Aktivität

Beim autonomen Planen arbeiten der Operator, der OBC und das APS, die bereits bei der Erläuterung der statischen Struktur (siehe Abbildung 6-2) eingeführt wurden, in einer Interaktion zusammen. Weiterhin sind indirekt das COM-Modul, die Kontroll- und die Bodenstation involviert. Im Mittelpunkt steht die zum OBC dazugehörende Liste mit den Kommandos (*Satellite Commands*) für den Satelliten. Über diese lässt sich das Verhalten des Satelliten bestimmen. Beim konventionellen Satellitenbetrieb (*control satellite*) kann der Operator die Liste per Telekommandos verändern. Das von der Bodenstation gesendete Telekommando wird über die Antenne und dem Receiver des COM-Moduls empfangen, dekodiert und dem OBC als Kommando zur weiteren Verarbeitung weitergereicht. Dort wird dieses nach einer Überprüfung (*verify command*) entweder sofort ausgeführt oder in die besagte Liste aufgenommen. Im ersteren Fall handelt es sich hierbei um ein sofortiges Kommando (*immediate command*), im letzteren Fall um ein zeitbegrenztes Kommando (*time-tagged command*), das erst zu einem spezifiziertem Zeitpunkt ausgeführt wird. Zum konventionellen Betrieb gehört ebenfalls der Datenverkehr in entgegen gesetzter Richtung. Hierbei werden sogenannte Housekeeping Daten über den Zustand des Satelliten und seiner Komponenten in regelmäßigen Zeitabständen erhoben und zwischengespeichert. Beim Kontakt mit der Bodenstation wird der Speicher ausgelesen und der Inhalt mit Hilfe des COM-Moduls als Telemetrie verpackt und zur Bodenstation gesendet.

Beim autonomen Betrieb ist eine Kommandierung nicht nur über das COM-Modul (und dadurch indirekt über den Operator) möglich, sondern auch über das APS. Das APS ist ein komplexes System. Es besitzt daher eine eigene zentrale Steuerung (*central control*), um seine internen Abläufe zu koordinieren. Es kann an ihn gerichtete Kommandos interpretieren, verifizieren und ausführen. Damit lässt sich sein Verhalten beeinflussen. In entgegen gesetzter Richtung erfasst es seinen internen Zustand (*generate housekeeping*) und sendet diese Information als Housekeeping an den OBC. Dieser ist auch in der Lage, das APS zu konfigurieren (*configurate*). Hierbei werden Parameter in einem Speicher verwaltet, mit denen sich das Verhalten des APS hinsichtlich bestimmter Kriterien einstellen lässt. Mit den oben beschriebenen Funktionalitäten sind die Anwendungsfälle (aus Abbildung 6-3) Kommandierung, Überwachung, Konfiguration und Steuerung abgedeckt.

Das autonome Planen (der verbliebene Anwendungsfall) ist ein komplexer Vorgang und daher in mehrere Teilfunktionen aufgegliedert. Es handelt sich dabei vom Prinzip her um ein Kontrollsystem. Eine der Aufgaben ist daher die Lageerfassung. Die Lage lässt sich zum einem aus den Housekeeping-Daten und zum anderen aus den aktiv gelisteten Kommandos des Satelliten einschätzen. Die Housekeeping-Daten beschreiben den aktuellen Zustand des Satelliten, während die Kommandos mit Hilfe geeigneter vorausschauender Methoden Aufschluss über die zukünftige Zustandsentwicklung geben können. Die Lageinformationen, bestehend aus ausgewählten Housekeeping-Daten und den gelisteten Kommandos, müssen daher mittels geeigneter Kommunikationswege zwischen dem OBC und dem APS übermittelt werden und in einem Speicher als Kopie im APS verwaltet werden. Um die Aktualität der Daten in der Kopie zu gewährleisten, muss zwischen ihr und den Originaldaten eine Synchronisierung stattfinden. Eine weitere Aufgabe des Kontrollsystems ist es, die erfasste Lage auszuwerten und notfalls gegenzusteuern, falls die aktuelle Lage vom Sollzustand abweicht. Auf das APS übertragen, bedeutet dies, dass eine Entscheidungsfindung (*decide*) notwendig ist, bei der festgelegt wird, welche Aktivitäten der Satellit zum aktuellen Zeitpunkt oder in der Zukunft ausführen soll. Das APS folgt dabei drei Paradigmen. Die Entscheidungsfindung wählt die Eingriffe (*Decissions*) sowohl zielorientiert, regelorientiert als auch ereignisorientiert aus. Abstrakt betrachtet kann ein Ereignis als das Eintreffen eines Signals aufgefasst werden. Das könnte zum Beispiel das Erreichen eines bestimmten Zeitpunktes sein, oder das Eintreffen von externen Daten. Regeln können als Konstrukte aufgefasst werden, die Aktivitäten anstoßen, wenn bestimmte Bedingungen erfüllt sind. Diese können boolesche Ausdrücke enthalten, die sich sowohl auf die erfasste Lage des Satelliten als auch auf die bereits erwähnten Signale beziehen. Die regelbasierte Entscheidungsfindung arbeitet in dem Sinne reflexartig. Dank der verwendeten booleschen Logik können Regeln schnell ausgewertet werden. Ziele geben, bezogen auf die Lage des Satelliten, einen gewünschten Zustand vor, der durch die Entscheidungsfindung erreicht werden soll. Sie sind daher deklarativ. Dabei werden Aktivitäten angestoßen. Die Konzepte Regeln,

Ereignisse und Ziele lassen sich auch beliebig miteinander kombinieren. Die Entscheidungsfindung kann die Ausführung von Aktivitäten nur vorschlagen. Deren Ausführung muss in einer nachfolgenden Ablaufplanung (*schedule*) geplant werden. Dabei müssen neben den von den Aktivitäten benötigten Ressourcen auch die Beschränkungen für die Abläufe berücksichtigt werden. Es kann daher nicht garantiert werden, dass alle von der Entscheidungsfindung vorgeschlagenen Eingriffe auch tatsächlich von der Ablaufplanung geplant werden können. Das Ergebnis der Ablaufplanung ist der Plan (*Schedule*). Es ist allgemein bekannt, dass die Algorithmen zur Ablaufplanung eine nichtpolynomielle Berechnungszeit haben (siehe Abschnitt 2.3). Die Ablaufplanung verhindert dadurch ein für die Behandlung kurzfristiger nicht-deterministischer Ereignisse (z.B. Naturphänomene wie Blitze oder Meteoreintritte) erforderliches, schnelles und reflexartiges Verhalten des Planungssystems und folglich des Satelliten. Aus diesem Grund kann in solchen Fällen die komplexe Ablaufplanung nach der Entscheidungsfindung übersprungen werden. Dies geschieht, indem die zuvor vorgeschlagenen Eingriffe (*Decissions*) direkt als geplante Aktivitäten zu vordefinierten Zeitpunkten in den Plan (*Schedule*) übernommen werden. Diese Vorgehensweise ist beherrschbar, da sie in polynomieller Zeit abgearbeitet werden kann. Sie hat jedoch einen Nachteil: Es kann passieren, dass die neuen Aktivitäten mit dem bereits bestehenden Plan ablauftechnisch nicht kompatibel sind. Dieser Konflikt muss durch weitere Maßnahmen im Rahmen einer vereinfachten Ablaufplanung gelöst werden. Dabei werden lediglich konkurrierende Aktivitäten aus dem Plan verworfen. Erst wenn ein gültiger Plan vorliegt, können die geplanten Aktivitäten, wie ursprünglich geschehen, weiter verarbeitet werden. Der nächste Schritt besteht darin, aus den geplanten Aktivitäten eine Sequenz von Kommandos (*satellite commands*) für den Satelliten abzuleiten und in einer separaten Liste zwischen zu speichern. An dieser Stelle soll erneut darauf hingewiesen werden, dass es sich bei der soeben genannten Liste nicht um die Kommandoliste vom OBC handelt. Stattdessen handelt es sich um einen separaten unabhängigen Speicherbereich innerhalb des APS. Alle Prozessschritte, von der Lageerfassung über die Ablaufplanung bis hin zur Übersetzung, finden im APS statt. Bevor die generierte Kommandosequenz letztendlich an den Bordcomputer weitergereicht wird, sollte sie nochmals in einer letzten Instanz überprüft werden. Damit soll sichergestellt werden, dass die generierten Kommandos den Satelliten nicht in Gefahr bringen. Um den Kontrollkreis zu schließen, müssen zuletzt noch die zwei Kommandolisten des APSs und des OBCs abgeglichen werden. Wie bereits beschrieben wurde, ist ein Synchronisierungsmechanismus (*synchronize satellite commands*) vorgesehen. Dieser führt laufend einen Abgleich zwischen den beiden Kommandolisten im APS und im OBC durch. An dieser Stelle sei anzumerken, dass die Synchronisationsvorgänge nicht trivial sind. Je nach Bordcomputer kann die Kommandoliste nur auf eine bestimmte Art und Weise manipuliert werden. So sind zum Beispiel manchmal Löschungen vereinzelter Kommandos aus der Liste nicht möglich. Stattdessen muss die Liste komplett gelöscht und unter Berücksichtigung der Änderungen neu beschrieben werden. Andere OBCs erlauben hingegen Modifikationen an vereinzelten Kommandos. Der Synchronisationsmechanismus muss daher flexibel einstellbar sein. Des Weiteren liefert der OBC im Rahmen der Synchronisierung Rückmeldungen (CMD FB). Damit kann das APS auf Abweichungen oder Fehler bei der autonomen Kommandierung reagieren. Mit der Synchronisierung als letzten Schritt wird der Kontrollkreis geschlossen und eine autonome Planung des Satelliten mit Hilfe des APSs ermöglicht.

Zuletzt soll noch erwähnt werden, dass die für die autonome Planung verwendeten Funktionalitäten bei ihrer Ausführung wahlweise auf gewisse Dienste zur Unterstützung zugreifen können:

- Zeit: Dieser Dienst verwaltet Informationen zur Zeit und kann bestimmte Berechnungen und Umrechnungen durchführen.
- History: Jeder Teilnehmer kann wahlweise seine Aktionen aufzeichnen lassen. Diese Informationen können vom Operator eingesehen werden. Dadurch kann dieser die internen Abläufe bei der autonomen Planung besser nachvollziehen.
- Orbit-Modell: Die Ablaufplanung arbeitet nur nach zeitlichen Kriterien. Bei der Entscheidungsfindung müssen jedoch häufig (z.B. bei Erdbeobachtungsmissionen) Orbitangaben ausgewertet und in zeitliche Größen umgerechnet werden. Zu diesem Zweck

bietet dieser Dienst bestimmte Berechnungen und Umrechnungen mit Hilfe diverser Orbit-Modelle (z.B. SGP4) an.
- Satellitenmodell: Dieser Dienst bietet eine Schnittstelle an, um das APS an verschiedene Satellitenmissionen anzupassen. Hier werden sozusagen die Möglichkeiten des Satelliten definiert. Intern verwaltet dieser Dienst Listen von Aktivitäten, Ressourcen, Beschränkungen, Kommandos, Housekeeping, Planungsziele, -regeln und -ereignisse.

Im Rahmen des ausführlichen Entwurfes (siehe Abschnitt 7) müssen folgende wichtige Fragestellungen beantwortet werden:

- Wie findet die Synchronisierung zwischen dem OBC und dem APS statt?
- Welche Algorithmen können für die Entscheidungsfindung und die Ablaufplanung der Aktivitäten verwendet werden?
- Wie ist der Operator in die autonome Planung eingebunden?
- Wie kann eine Relation zwischen den Aktivitäten und den Kommandos hergestellt werden?
- Was passiert, wenn die Entscheidungsfindung die Ablaufplanung übergeht?
- Wie kann die Sicherheit des Satelliten gewährleistet werden?

Zum Schluss sollen noch einige wichtige Punkte genannt werden, durch die sich dieses Konzept positiv vom aktuellen Stand der Technik und der Forschung abgrenzt:

1. Eine der großen Stärken des APS ist es, die Ablaufplanung in zeitkritischen Fällen zu umgehen. Dadurch lassen sich deutliche Geschwindigkeitsvorteile gewinnen. Die Laufzeit der hieraus resultierenden Planung wird polynomiell. Es ist nämlich die Ablaufplanung, die ein nicht-polynomielles Laufzeitverhalten aufweist und somit mit einen großen Rechenaufwand verbunden ist. In der Literatur-Recherche (siehe Abschnitt 4) wurde festgestellt, dass autonom arbeitende Planungssysteme meistens die Ablaufplanung starr an die Entscheidungsfindung koppeln und dadurch in dieser Hinsicht unflexibel sind.
2. Aufgrund seiner Einstellungsmöglichkeiten (z.B. Regeln, Aktivitäten, Beschränkungen) ist das APS vielseitig einsetzbar und nicht an einzelne Satellitenmission gebunden. Diese Flexibilität ist beim aktuellen Stand der Technik nicht gegeben, da die derzeitigen autonomen Systeme anscheinend jeweils speziell für eine einzelne Satellitenmission entworfen wurden.
3. Ein weiteres Unterscheidungsmerkmal ist die tiefgreifende Konfigurierbarkeit des APS per Kommandierung im Orbit, wodurch sogar eine komplette Verhaltensänderung zugelassen wird. Dieser Grad an Freiheit konnte in der Recherche nicht festgestellt werden. In der Regel sind kleinere Anpassungen im Onlinebetrieb, meist durch eine Änderung der Zielvorgaben, möglich, jedoch nicht am Grundverhalten.
4. Bisher konnten in der Recherche keine Hinweise auf ein autonomes Planungssystem, die mit den Beschränkungen einer Kleinstsatellitenmission (siehe Abschnitt 1.6), insbesondere hinsichtlich des Bedarfs an elektrischer Leistung und Energie, vereinbar wären, gefunden werden. Durch ein frühzeitiges Einbeziehen dieser Randbedingungen in den Entwicklungsprozess (siehe Abschnitt 7. Realisierung) des APS kann dadurch ein Vorstoß in diese Richtung unternommen werden. Die Einschränkungen sind zum einen bei der Auswahl der zur Verfügungen stehenden Hardware-Komponenten für die Plattform, auf der das APS betrieben werden soll, und zum anderen bei der Gestaltung einer entsprechend Ressourcen schonenden Software, deutlich bemerkbar.

7 Realisierung

Dieses Kapitel zeigt einen möglichen Lösungsvorschlag, wie die in Abschnitt 1.3 beschriebene Fragestellung des autonomen Planungsproblems gelöst werden könnte, auf. Der Vorschlag stellt somit den Hauptbeitrag des Autors zur Forschung autonomer Systeme dar. Die Forschung ist in ihrer Gesamtheit noch nicht abgeschlossen. Vielmehr handelt es sich bei den in dieser Dissertation zusammengefassten Erkenntnissen um Zwischenergebnisse auf dem Weg zu vollautonomen Systemen.

Die Forschungsarbeiten des Autors orientieren sich an sogenannten Referenzobjekten und Referenzszenarien für mögliche Satellitenmissionen, zur deren Durchführung eine Autonomie benötigt wird bzw. sinnvoll erscheint. Sämtliche relevante Szenarien, sowie der Begriff selbst, werden in einem gesonderten Unterabschnitt erläutert. Sie bilden die Ausgangsbasis für die eigentliche Forschungsarbeit des Autors. Zu dieser gehören im Wesentlichen der Entwurf, die Umsetzung und der Test technischer Systeme mit autonomen Fähigkeiten. Insgesamt wurden drei solcher Systeme erstellt. Sie werden jeweils in einem eigenen Unterabschnitt behandelt:

- APS (Autonomes Planungssystem)
- ASAP (Autonomer Sensor, Autonomer Planer)
- ATUS (Testumgebungssystem für APS und ASAP)

Das APS ist ein reines, aus mehreren Teilkomponenten bestehendes, Softwaresystem. Es bietet autonome Dienste für die Bordsteuerung des Satelliten an und ist portierbar, d.h. es ist an keine spezifische bzw. dedizierte Hardware gebunden.

Zur Realisierung der Referenzszenarien wurde ASAP entwickelt. Dieses System dient als Hardware-Plattform und setzt mitunter zur Erfüllung seiner (autonomen) Aufgaben das APS ein. Daneben beinhaltet die Plattform zusätzlich einen schnellen Bildsensor samt Optik und elektronischer Auswertung. Im Rahmen dieser Dissertation werden diese optischen Merkmale von ASAP zwar aufgegriffen (siehe Abschnitt 7.3.5.1), aus Sicht des APS dient ihm ASAP aber primär als eine reine Hardware-Plattform, auf der seine Software ausgeführt wird. Die hierin enthaltenden Komponenten und Algorithmen zur Erfüllung autonomer Funktionalitäten stellen den Schwerpunkt dieser Dissertation dar.

Da es sich beim APS und bei ASAP um komplexe Systeme handelt, müssen deren Funktionsfähigkeit (insbesondere die autonomen Funktionen) in besonderen Verfahren getestet werden. Zu diesem Zweck wurde das Testumgebungssystem ATUS entwickelt. Hierbei handelt es sich sowohl um ein HIL (Hardware-In-The-Loop) - als auch um ein SIL (Software-In-The-Loop) - Testsystem. Als HIL - Testsystem wurde es verwendet, um für ASAP die Schnittstelle zur Satellitenplattform (d.h. der Satellitenbus) zu emulieren, als SIL – Testsystem wurde es verwendet, um mittels Rapid-Prototyping-Techniken schnell neue Softwarekomponenten (z.B. autonome Algorithmen) zu erstellen und zu testen. Hierdurch lassen sich wertvolle Erkenntnisse erzielen, die beim Vorhaben SONATE in zukünftige Entwicklungen einfließen sollen. Bei diesem Satelliten soll ASAP, anstatt mit dem Testsystem ATUS, nun mit dem eigentlichen zu entwickelnden Satellitenbus verbunden werden. Die nachfolgende Abbildung verdeutlicht nochmals die Beziehung der einzelnen Systeme zueinander:

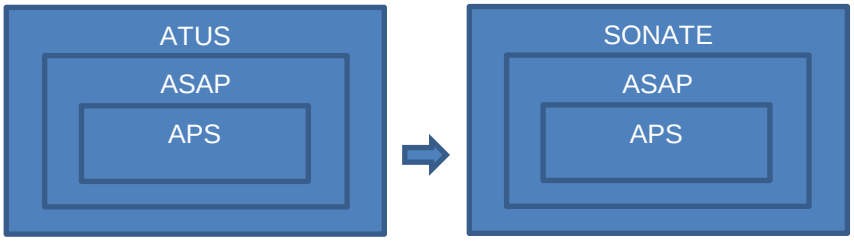


Abbildung 7-1: Hierarchische Übersicht der Systeme

7.1 Referenzobjekte und Referenzszenarien

In der Raumfahrt beschreibt und veranschaulicht ein „Referenzszenario“ bestimmte Aspekte einer Satellitenmission. Es dient überwiegend der Planung. Im vorliegenden Fall wurde insbesondere der Aspekt „Autonomie“ untersucht. Im Vordergrund steht dabei die Frage: Was soll das System (z.B. der Satellit, oder eines seiner Teilsysteme) vollbringen? Hierbei wird der Kontext der Systeme als Ganzes betrachtet. Technische Details und Implementierungen rücken in den Hintergrund und werden erst in den nachfolgenden Abschnitten im Detail betrachtet.

Wie in der Einleitung im Abschnitt 1.3 der vorliegenden Dissertation bereits erwähnt wurde, liegt der Forschungsschwerpunkt der Arbeitsgruppe, in der der Autor tätig ist, unter anderem in der höheren Autonomie im Satellitenbetrieb und in der Erforschung unbekannter Himmelsphänomene. Insbesondere stehen im Mittelpunkt Phänomene, Ereignisse und besondere Objekte, die sich entweder direkt auf der Erdoberfläche oder in Erdnähe (Entfernung zur Erdoberfläche < 1000 Km) ereignen. Diese Vorkommnisse werden in der Dissertation fortan als „Gegenstände der wissenschaftlichen Untersuchung“ oder auch schlicht als „Gegenstände“ zusammengefasst und bezeichnet. Die Entwicklung der autonomen Systeme orientiert sich generell an ausgewählten Exemplaren. Diese werden in Zusammenhang mit den Referenzszenarien auch als Referenzobjekte benannt und in den wissenschaftlichen Aufsätzen [1] und [2] erstmals eingeführt und erläutert. Zu den wissenschaftlichen Gegenständen gehören auszugsweise:

- Phänomene
  - Meteore und Fireballs
  - Sprites und Blue Jets
  - Blitze
- Ereignisse
  - Vulkanausbrüche
  - Überschwemmungen
  - Großflächenbrände
- Objekte
  - Weltraumschrott (Space Debris)
  - Satelliten und Stationen (z.B. ISS)

Das Ziel der Forschung liegt darin, Erkenntnisse über diese Gegenstände zu gewinnen. Dazu müssen Daten mittels Beobachtung gesammelt werden. Solche Beobachtungen können mit Hilfe von Satelliten und den darin integrierten Sensoren realisiert werden. Einsatzmöglichkeiten werden in Referenzszenarien skizziert. Diese beschreiben, welche Schritte bei der Missionsplanung für den Satelliten generell notwendig sind. Typischerweise werden dabei die Gegenstände mit den auf einem Satelliten verbauten Sensoren (z.B. einer Kamera oder Teleskop) detektiert, aufgezeichnet und, falls möglich, nachverfolgt (d.h. im Blickfeld der optischen Geräte behalten, was eine Lageregelung des Satelliten bedingt). Durch die Nachverfolgung kann eine Videosequenz erzeugt werden. Im Anschluss sollen die gewonnen Daten zur Bodenstation übertragen werden, wo eine wissenschaftliche Auswertung erfolgt.

Die Auslegung der Referenzszenarien erfolgt anhand der zu beobachtenden Referenzobjekte. Diese unterscheiden sich hinsichtlich ihrer zeitlichen Dauer und ihrer zeitlichen Bestimmbarkeit:

- Zeitliche Bestimmbarkeit
  - Deterministisch (z.B. Ort eines Objektes auf einem Orbit)
  - Nichtdeterministisch (z.B. Blitze, Meteoreintritt)
- Zeitliche Dauer
  - Kurzfristig (z.B. Blitz), Zeitraum: Millisekunden, Sekunden
  - Mittelfristig (z.B. Meteoreintritt), Zeitraum: Sekunden, Minuten
  - Langfristig (z.B. Vulkanausbruch), Zeitraum: Minuten, Stunden, Tage

Die meisten Referenzszenarien setzen das Vorhandensein einer Autonomie voraus. Dies ist insbesondere der Fall, je kürzer die Dauer und je stärker der Aspekt der zeitlichen Unbestimmbarkeit gegeben ist. Die Gründe hierfür wurden bereits im Abschnitt 1.3 Problemstellung erläutert.

In Betracht kommende Referenzszenarien müssen aus einer Abfolge von Einzelschritten entworfen werden:

- Beobachten
  Der Satellit sucht mit seinen Sensoren den Bereich im Raum ab, in dem die Referenzobjekte vermutet werden, bzw. am wahrscheinlichsten auftreten. Evtl. angesammelte Nutzdaten, insbesondere Suchergebnisse, werden onboard zwischengespeichert.
- Nachverfolgen
  Der Satellit hat einen wissenschaftlichen Gegenstand detektiert und versucht, diesen so lange wie möglich im Sichtbereich seiner Sensoren zu halten. Dabei anfallende Nutzdaten werden ebenfalls onboard zwischengespeichert. Bei der Nachverfolgung muss in der Regel das Lagekontrollsystem des Satelliten in die autonome Planung miteinbezogen werden.
- Aufladen
  Der Satellit richtet seine Solarpanele in Richtung Sonne aus, um seine elektrische Ladung in den Batterien aufzufrischen.
- Downlink
  Der Satellit ist in Kommunikationsreichweite mit einer ihm zugeordneten Bodenstation und versucht, seine gesammelten Daten (wissenschaftliche Nutzdaten und Housekeeping) zur Erde zu senden.

Im Folgenden sollen nun einige potentielle Referenzszenarien vorgestellt werden. Es sei noch anzumerken, dass die folgenden Referenzszenarien nicht isoliert betrachtet werden dürfen. Vielmehr können sie auch gleichzeitig stattfinden, bzw. sich überlappen. Ferner muss berücksichtigt werden, dass Referenzszenarien evtl. noch durch autonome Planungsalgorithmen angepasst werden, damit auf nicht vorhersehbare Umstände reagiert werden kann (Beispiel: Zusätzliche Ladezyklen für den Satelliten müssen vorgesehen werden).

<u>Referenzszenario für kurzfristige und nichtdeterministische Referenzobjekte (z.B. Blitze)</u>

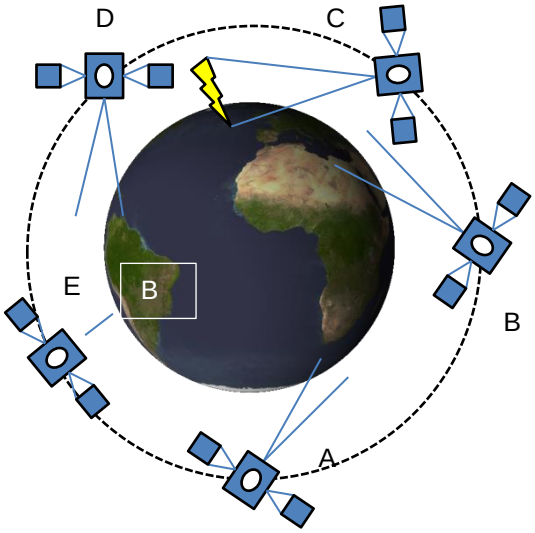


Abbildung 7-2: Ablauf für kurzfristige Beobachtungen

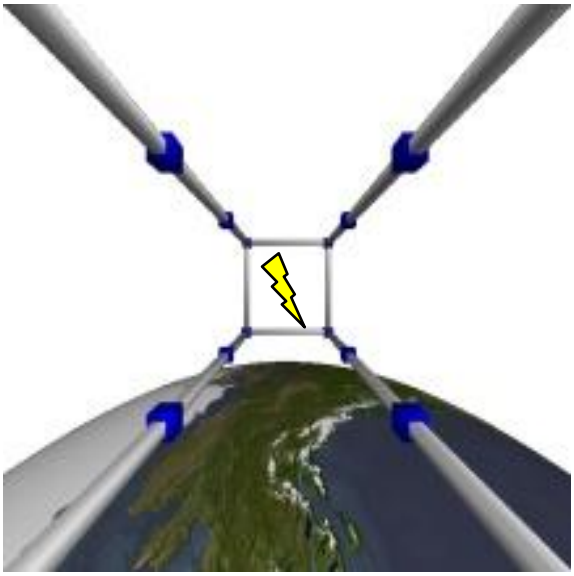


Abbildung 7-3: Darstellung aus der Sicht des Sensors

Wegen ihrer geringen Dauer und ihres nichtdeterministischen Auftretens können die Ziele dieser Beobachtungsgruppe (z.B. Blitze) nur detektiert werden, indem die Orte ihres Erscheinens nach

Möglichkeit permanent mit Sensoren überwacht werden. Der Ablauf kann wie in der obigen Abbildung 7-2 dargestellt stattfinden.

Mit dem Beginn der Beobachtung (A) wird der Sensor (z.B. die Kamera) auf das Zielgebiet (siehe Abbildung 7-3, in diesem Fall die Erdatmosphäre als Ausgangsort der Blitze) ausgerichtet und permanent nachgeführt (B). Die Nachführung erfordert die Einbindung des Lageregelungssystems (ACS) des Satelliten. Solange vom Sensor keine relevanten Ereignisse, Phänomene oder Objekte erkannt werden, werden die aufgezeichneten Daten wieder verworfen. Dabei kann der Satellit mehrfach die Erde umkreisen, bis ein interessantes Ereignis (z.B. ein Blitz) eintritt (C). In diesem Fall werden die aufgezeichneten Daten nicht verworfen, sondern zwischengespeichert. Die Beobachtung wird im Anschluss wieder fortgesetzt (D), um weitere potentielle Ereignisse zu detektieren. Sobald der Satellit in den Kommunikationsbereich seiner Bodenstation (BS) eintritt (E), sendet er seine aufgezeichneten Daten zur Erde. Dort findet dann die weitere wissenschaftliche Auswertung statt.

Referenzszenario für mittelfristige und nichtdeterministische Referenzobjekte (z.B. Meteoreintritte)

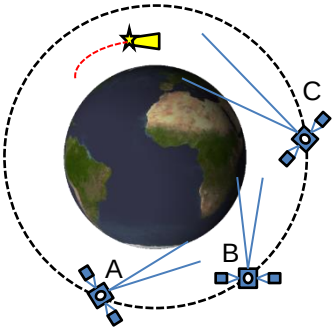


Abbildung 7-4: Detektierung

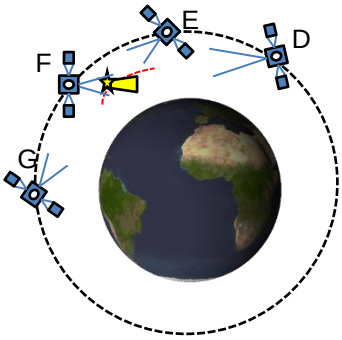


Abbildung 7-5: Aufnahme und Nachverfolgung

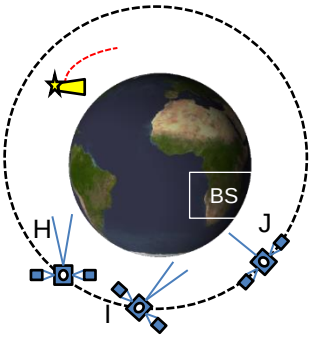


Abbildung 7-6: Abschluss

Um bei mittelfristigen, nichtdeterministischen Beobachtungen (z.B. Meteoreintritt inkl. Verbrennung in der Erdatmosphäre) die Menge der zu gewinnenden wissenschaftlichen Daten zu erhöhen, muss das Lageregelungssystem in die autonome Planung des Satelliten miteingebunden werden. Dies ist notwendig, um die verwendeten Sensoren des Satelliten solange wie möglich auf das zu beobachtende Referenzobjekt auszurichten. Diese Vorgehensweise ähnelt dem vorherigen Referenzszenario für kurzfristige Beobachtungen. Die Schritte A-C (siehe Abbildung 7-4) bis zur Detektierung sind identisch. Im Anschluss findet die Nachverfolgung (Schritte D – G in Abbildung 7-5) statt. Diese wird solange fortgeführt, bis der Satellit auf seiner Bahn den Sichtbereich des Referenzobjektes verlässt, oder dieses aufhört zu existieren. Wie bereits zuvor bei kurzfristigen Beobachtungen wird nach Abschluss der Nachverfolgung (Schritt H, in Abbildung 7-6) nach weiteren Referenzobjekten gesucht (Schritt I). Auch müssen bei Gelegenheit die gewonnenen wissenschaftlichen Daten zur Bodenstation übermittelt werden (Schritt J). Es ist davon auszugehen, dass das Referenzobjekt nicht mehr existiert, wenn der Satellit die Erde umrundet hat und sich wieder am Ausgangsort der Beobachtung (ursprünglich Schritt C) befindet.

Referenzszenario für langfristige und nichtdeterministische Referenzobjekte (z.B. Vulkanausbrüche)

Langfristige, nichtdeterministische Beobachtungen (z.B. Vulkanausbrüche) unterscheiden sich von mittelfristen, nichtdeterministischen Beobachtungen dadurch, dass sich bei ihnen aufgrund der Dauer die Möglichkeit bietet, die wissenschaftliche Erfassung nach einer Umrundung des Satelliten um die Erde wieder aufzunehmen. Die Schritte A – K, dargestellt in den Abbildung 7-7 und Abbildung 7-8, ähneln daher denen von mittelfristigen, nichtdeterministischen Beobachtungen. Der Unterschied zu diesen besteht darin, dass die wissenschaftliche Aufnahme des Referenzobjektes mit den Sensoren, für die Dauer seiner Existenz, beliebig oft wiederholt werden kann. Auch hierbei müssen bei

Gelegenheit die gewonnenen wissenschaftlichen Daten zur Bodenstation (Schritte F und L) übermittelt werden.

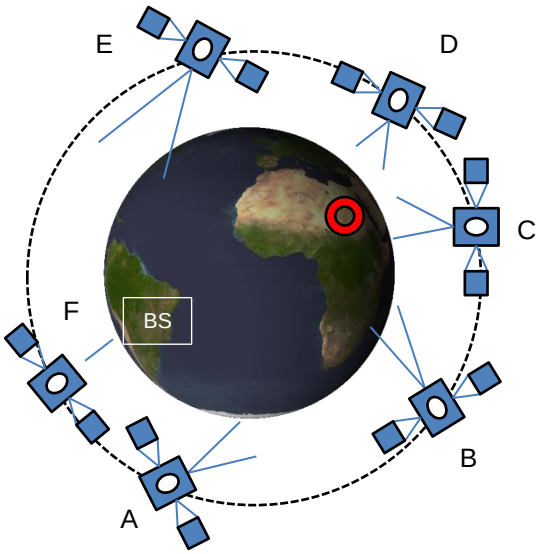


Abbildung 7-7: Detektierung, Nachverfolgung und Übertragung

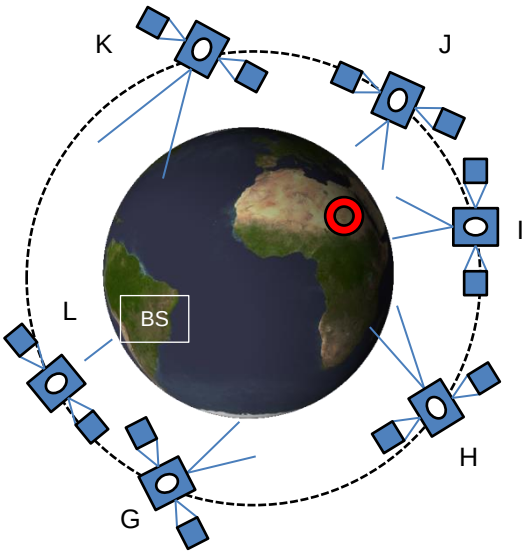


Abbildung 7-8: Wiederaufnahme, Nachverfolgung und Übertragung

<u>Referenzszenario für kurzfristige und deterministische Referenzobjekte (z.B. Erdaufnahmen)</u>

Besonders im Bereich der Erdbeobachtung werden häufig Bildaufnahmen bestimmter Gebiete auf der Erde zu einer bestimmten Tageszeit angefordert. Ein mögliches Szenario könnte so aussehen, dass der Satellit eine Liste mit derartigen Aufträgen verwaltet und selbstständig einen Plan entwickelt, wann diese Aufträge abgearbeitet werden sollen. Dabei sollen zusätzliche Beschränkungen wie Deadlines für Bildaufnahmen oder bestimmte Winkelpositionen, die das aufnehmende Instrument in Richtung Zielgebiet einnehmen muss, berücksichtigt werden.

Der Ablauf erfolgt dann auf die Weise, wie er in Abbildung 7-9 dargestellt ist.

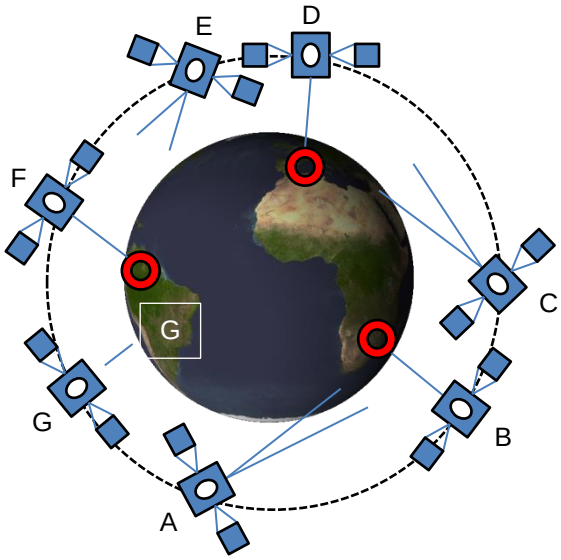


Abbildung 7-9: Aufnahme von Gebieten auf der Erde

Das Szenario beginnt damit, dass der Satellit zuerst nach nichtdeterministischen Referenzobjekten Ausschau hält (Schritt A), da er sonst keine andere sinnvolle Aufgabe hat. Sobald ein Zielgebiet des Satelliten in Reichweite gelangt, wird er autonom seine Instrumente hierauf ausrichten (Schritte B, D und F) und die Aufnahmen tätigen. Sobald die Aufgabe erledigt ist, sucht der Satellit wieder nach nichtdeterministischen Referenzobjekten (Schritt C und E). Bei sich bietender Gelegenheit werden schließlich die aufgenommenen Daten zur Erde übermittelt (Schritt G).

Referenzszenario für langfristige und deterministische Referenzobjekte (z.B. fremde Satelliten)

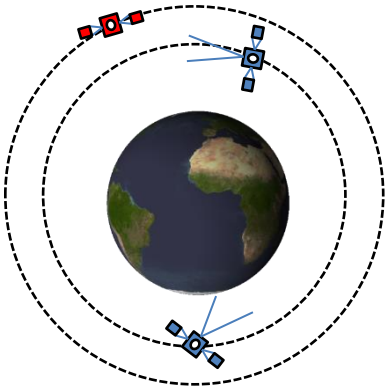


Abbildung 7-10: Beobachtung und Nachverfolgung

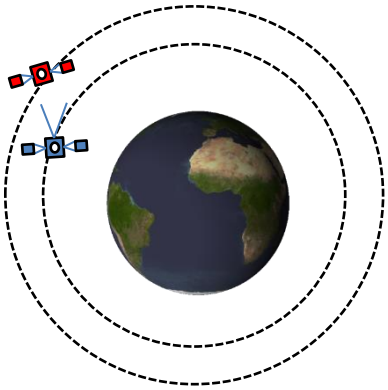


Abbildung 7-11: Nachverfolgung

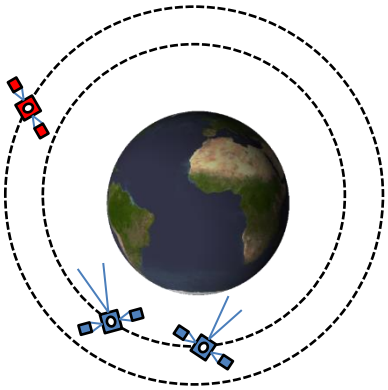


Abbildung 7-12: Nachverfolgung und Beobachtung

Der Vorteil von deterministischen Referenzobjekten besteht darin, dass jederzeit berechnet werden kann, wann und wo sie auftreten. Dazu werden in diesem Beispiel zur Beobachtung eines anderen Satelliten Bahnmodelle (z.B. Kepler oder SGP4) benötigt. Zunächst beobachtet der Satellit wie bei den anderen Referenzszenarien die Erdatmosphäre auf der Suche nach Naturphänomenen. Sobald der Zielsatellit in den Sichtbarkeitsbereich der eingesetzten Sensoren kommt, was mit Hilfe des Bahnmodels berechnet werden kann, schaltet der Satellit auf Nachverfolgung um, und hält diesen Zustand solange wie möglich aufrecht (siehe Abbildung 7-11). Bei Verlust des Sichtkontaktes wechselt der Satellit anschließend wieder in den Beobachtungsmodus (siehe Abbildung 7-12), gefolgt von einem Datenaustausch mit der Bodenstation. Das Szenario kann bei neuem Sichtkontakt wiederholt werden.

7.2 APS – Autonomes Planungssystem

Dieser Abschnitt beschäftigt sich mit dem Autonomen Planungssystem (APS). Es handelt sich hierbei um ein reines Softwaresystem, das mit der in Abschnitt 5 beschriebenen Methode entwickelt wurde. Die nachfolgenden Unterabschnitte beschreiben die einzelnen Zwischenschritte des Entwicklungsprozesses.

7.2.1 Zielstellung

Die primäre Aufgabe des APS besteht in dem Erzeugen und Umsetzten autonomer Entscheidungen. Hierbei muss das zu entwickelnde System mindestens so viel Flexibilität aufweisen, dass sich die in Abschnitt 7.1 dargestellten Referenzszenarien realisieren lassen. Aspekte der Hardware bleiben dabei unberücksichtigt. Sie werden im Abschnitt 7.3 ASAP behandelt, wenn das APS auf die Plattform portiert wird. An dieser Stelle ist bereits erkennbar, dass die Aufgabe nicht allein vom APS gelöst werden kann. Vielmehr muss dieses im Verbund mit anderen Softwareanwendungen des Satelliten (z.B. Manager für Kommandos und Housekeeping) arbeiten, um die Satellitenmission erfolgreich durchführen zu können.

7.2.2 Stakeholder

Die nachfolgende Abbildung stellt eine Übersicht über alle Stakeholder dar, die eine relevante Beziehung zum APS haben.

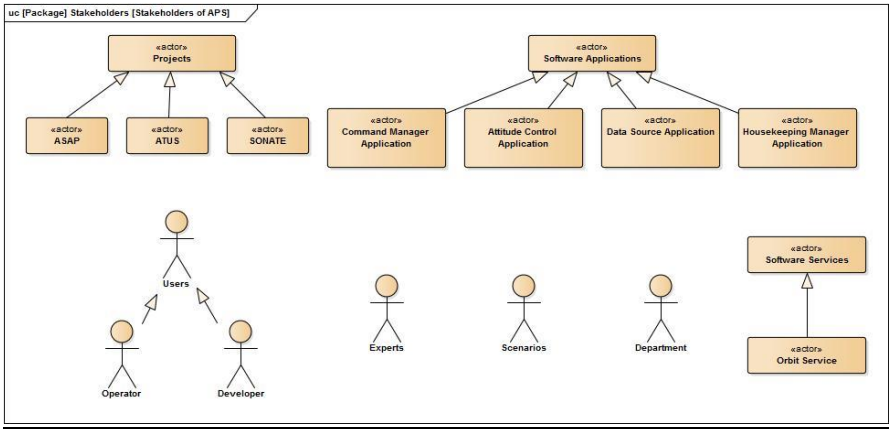


Abbildung 7-13: Stakeholders von APS

Wie bereits erwähnt wurde, arbeitet das APS im Verbund mit anderen Softwareanwendungen.

Dazu gehören:

- Kommandoverwaltung (Command Manager Application)
  Indem die Kommandosequenzen in der Kommandoverwaltung manipuliert werden, kann das APS das Verhalten des Satelliten autonom ändern.
- Zustandserfassungsverwaltung (Housekeeping Manager Application)
  Das APS benötigt Informationen über den Zustand des Satelliten und seiner Komponenten.
- Lageregelung (Attitude Control Application)
  Eine Interaktion mit dieser Anwendung ist notwendig, wenn die Ausrichtung des Satelliten im Raum, wie in den Referenzszenarien beschrieben, geändert werden muss.
- Datenquellen (Data Source Application)
  Das APS benötigt irgendeine Form von Dateninput (z.B. Meta-Daten), auf dessen Grundlage autonome Entscheidungen für den Satelliten getroffen werden sollen.

Zur Bewältigung typischer Aufgabenstellungen, wie sie insbesondere bei den im Abschnitt 7.1 beschriebenen Referenzszenarien auftreten, benötigt das APS die Dienste eines Softwaremoduls zur Orbit- und Bahnberechnung. Die von ihm bereitgestellten Informationen sind notwendig, wenn der Aufenthaltsort von Bahnobjekten (z.B. Referenzobjekte, aber auch der eigene Satellit) in der Zukunft benötigt wird.

Ferner treten menschliche Personen und Gruppierungen als Stakeholder auf. Als direkter Anwender wäre der Operator zu nennen. Dieser muss die autonomen Satellitenmissionen ausführen und überwachen. Der Autor selbst, dessen Lehrstuhl sowie die Arbeitsgruppe, der der Autor angehört, setzen ebenfalls Präferenzen. Des Weiteren beeinflussen Experten (z.B. Prof. Montenegro) den Entwicklungsprozess, indem sie mit ihrem Fachwissen und Erfahrungen zum Vorhaben beitragen.

Zuletzt wären noch die Vorhaben ASAP, ATUS und SONATE als Stakeholder zu nennen. Im Rahmen dieser Vorhaben fanden die Entwicklungsarbeiten am APS und an den mit ihm (direkt und indirekt) interagierenden Partnern (z.B. Kommandoverwaltung), statt.

7.2.3 Anforderungen

Eine Analyse der Beziehungen und Abhängigkeiten zwischen dem zu entwickelnden autonomen Planungssystem (APS) und seinen Stakeholdern hat einen Satz von Nutzeranforderungen und Systemanforderungen – untergliedert in funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen – aufgezeigt. Sie werden in den nachfolgenden Tabellen aufgeführt. Die Anforderungen lassen sich zu inhaltlichen Gruppen, die jeweils einem Stakeholder zugeordnet sind, zusammenfassen. Elementare Anforderungen werden nach Möglichkeit begründet.

Die nachfolgende Tabelle 7-1 listet die Nutzeranforderungen an das APS

ID	Name und Begründung
1	Nutzeranforderungen
1.1	Anforderungen seitens des Operators
1.1.1	Das APS soll den Satelliten in die Lage versetzen, autonom zu handeln - angefangen bei der Entscheidungsfindung über die Planung bis hin zur Ausführung.
1.2	Anforderungen seitens des Entwicklers (Autor)
1.2.1	Das APS soll im laufenden Betrieb konfigurierbar und anpassbar sein. Dieses Leistungsmerkmal grenzt das APS vom aktuellen Stand der Technik ab (siehe Abschnitt 4.12)
1.2.2	Das APS soll mit den zur Verfügung stehenden Ressourcen (z.B. Masse, Energie, Leistung, Volumen, Speicher) einer Kleinstsatellitenmission (Nanosatellit) betrieben werden können.
1.7	Anforderungen seitens der Referenzszenarien (bzw. auch des Lehrstuhls)
1.7.1	Mit dem APS sollen sich die in Abschnitt 7.1 genannten Referenzszenarien durchführen lassen.

Tabelle 7-1: Nutzeranforderungen an das APS

Die nachfolgende Tabelle 7-2 listet die funktionalen Anforderungen an das APS

ID	Name und Begründung
2	Systemanforderungen (funktionale Anforderungen)
2.1	Anforderungen seitens des Operators
2.1.1	Daten Der Operator am Boden soll mit Daten versorgt (Verzögerung erlaubt) werden, die

	ihm Aufschluss über den Zustand des APS liefern. Von besonderem Interesse sind der aktuelle Plan für die Satellitenmission sowie die Datengrundlage für bereits getroffene Entscheidungen seitens des APS.
2.2	Anforderungen seitens des Entwicklers (Autor)
2.2.1	Konfigurierbarkeit Das APS soll an verschiedene Satellitenmissionen (Szenarien) anpassbar sein.
2.2.1.1	Anpassbarkeit Das APS soll anpassbar sein, ohne dass der Quellcode neu kompiliert werden muss.
2.2.1.2	Im Orbit Das APS soll auch anpassbar sein, wenn sich der Satellit im Orbit befindet.
2.3	Anforderungen seitens des Lehrstuhls
2.3.1	Betriebsmodi
2.3.1.1	Off Das APS muss so eingestellt werden können, dass es keinen Einfluss auf den Satelliten ausübt, so dass dieser auf konventioneller Art betrieben werden kann.
2.3.1.2	Passiv Das APS muss so eingestellt werden können, dass es zwar Zustandsdaten vom Satelliten auswertet und somit autonom Entscheidungen und Pläne berechnet, diese aber nicht zur Ausführung an den Bordcomputer des Satelliten weiterreicht, sondern nur zur Evaluierung an den Operator sendet.
2.3.1.3	Aktiv Das APS muss so eingestellt werden können, dass es Zustandsdaten vom Satelliten auswertet und somit autonom Entscheidungen und Pläne berechnet, diese dann auch zur Ausführung an den Bordcomputer des Satelliten weiterreicht.
2.3.2	Datenformate
2.3.2.1	TC Format Es muss berücksichtigt werden, dass für die Funkübertragung zur Kommandierung des Satelliten ein speziell vom Lehrstuhl entwickeltes Format verwendet werden soll.
2.3.2.2	TM Format Es muss berücksichtigt werden, dass für die Funkübertragung zur Telemetrieerfassung des Satelliten ein speziell vom Lehrstuhl entwickeltes Format verwendet werden soll.
2.4	Anforderungen seitens der Vorhaben/Projekte
2.5	Anforderungen seitens der Kommandoverwaltung
2.5.1	Das APS muss sich an das Protokoll der Kommandoverwaltung halten.
2.6	Anforderungen seitens der Zustandserfassungsverwaltung
2.6.1	Das APS muss sich an das Protokoll der Zustandserfassungsverwaltung halten.
2.7	Anforderungen seitens der Referenzszenarien
2.7.1	Autonome Planung Das APS muss über autonome Funktionen verfügen, damit die Referenzszenarien ausgeführt werden können.
2.7.1.1	Entscheidungen Das APS muss eigenständig Entscheidungen zur weiteren Missionsplanung des Satelliten treffen. Insbesondere muss entschieden werden, welche Aufgaben der Satellit in Zukunft wahrnehmen soll.
2.7.1.2	Ablaufplanung Das APS muss eigenständig berechnen, in welchen Zeiträumen die vom Satelliten zu tätigenden Arbeiten ausgeführt werden sollen. Dabei müssen Ressourcen und Beschränkungen des Satelliten berücksichtigt werden. Als Ergebnis muss ein gültiger Missionsplan ausgegeben werden.
2.7.1.3	Steuerung Das APS muss in der Lage sein, seine berechneten Missionspläne mit der Kommandoverwaltung abzustimmen und die Ausführung zu veranlassen.
2.7.1.4	Meta Daten Das APS muss in der Lage sein, Meta Daten von beliebigen Datenquellen zu empfangen und auszuwerten. Auf der Basis dieser Daten soll das APS seine autonomen Entscheidungen treffen.

2.7.2	Dienst zur Berechnung von Orbitdaten Das APS muss Informationen zur Position eines beliebigen Objektes (auch des eigenen Satelliten) im Orbit entweder eigenständig berechnen oder sich anderweitig beschaffen können.

Tabelle 7-2: Systemanforderungen (funktionale Anforderungen) an das APS

Die nachfolgende Tabelle 7-3 listet die Qualitätsanforderungen an das APS.

ID	Name und Begründung
3	Systemanforderungen (Qualitätsanforderungen)
3.7	Anforderungen seitens der Referenzszenarien
3.7.1	Entscheidungen (siehe funktionale Anforderung 1.7.1.1) mit Dringlichkeit müssen innerhalb einer festdefinierten Zeitschranke umgesetzt werden.

Tabelle 7-3: Systemanforderungen (Qualitätsanforderungen) an das APS

Die nachfolgende Tabelle 7-4 listet die Randbedingungen an das APS

ID	Name und Begründung
4	Systemanforderungen (Randbedingungen)
4.2	Anforderungen seitens des Entwicklers (Autor)
4.2.1	Das APS soll mit den zur Verfügung stehenden Ressourcen (z.B. Masse, Energie, Leistung, Volumen, Speicher) einer Kleinstsatellitenmission (Nanosatellit) betrieben werden können.
4.4	Anforderungen seitens der Vorhaben/Projekte
4.4.1	RODOS Das APS soll als eine Anwendung für eine RODOS-Plattform entwickelt werden, da davon auszugehen ist, dass es in einem Netzwerk mit RODOS-Knoten (z.B. OBC, VIDANA) eingebettet wird.

Tabelle 7-4: Systemanforderungen (Randbedingungen) an das APS

7.2.4 Systemkontext

In den vorherigen Abschnitten wurden die Stakeholder identifiziert und die Anforderungen an das APS analysiert. Auf der Grundlage dieser Vorarbeiten kann nun im nächsten Schritt der Kontext des zu entwickelnden Systems abgegrenzt werden (siehe Abbildung 7-14). Damit wird zum einen eine erste strukturierte Ordnung zwischen den beteiligten Systemen eingeführt, zum anderen wird jetzt klarer ersichtlich, welche Teilnehmer direkt miteinander interagieren und somit eine gemeinsame Schnittstelle aufweisen müssen.

Der Kontext stellt eine Sicht auf Anwendungsebene dar und wurde in die zwei Bereiche Bodensegment (*Ground Segment)* und Raumsegment (*Space Segment)* aufgeteilt. Letzterer stellt die unmittelbare Umgebung des APSs dar. Sämtliche Anwendungen des Raumsegments kommunizieren über die RODOS-Middleware und übertragen daher ihre Daten untereinander über RODOS-Topics. Der Großteil des autonomen Verhaltens wird durch die Interaktionen zwischen dem autonomen Planungssystem (*APS Application*), der Kommando-Anwendung (*Command Application*) und der Telemetrie-Anwendung (*Telemetry Application*) realisiert. Der Verbund dieser drei Systeme bildet das Grundgerüst einer Steuerschleife. Die Telemetrie-Anwendung hat unter anderem die Aufgabe, Zustandsdaten (Housekeeping) des Satelliten zu erfassen. Diese können vom APS über einen speziellen Synchronisierungsmechanismus (*Sync Hk Request, Sync Hk Answere*) abgerufen werden. Je nach Anwendungsfall kann eine weitere Form von Daten, sogenannte Meta-Daten, in die autonome Planung einfließen. Diese werden von speziell dafür entwickelten Anwendungen (*Meta Data Source Application*) bereitgestellt. Das APS generiert mit Hilfe seiner autonomen

Planungsfähigkeiten einen Plan und lässt ihn durch die Kommando-Anwendung (*Command Application*) mittels Kommandierung am Satelliten umsetzen. Dieser Vorgang setzt ebenfalls, wie bei der Zustandserfassung, einen Synchronisierungsmechanismus (*Sync Cmd*) voraus. Mit Feedback-Botschaften (Sync *Cmd Fb*) wird das APS informiert, ob die von ihm geforderten Änderungen am Missionsplan erfolgreich sind. Damit schließt sich der Steuerkreis. Zu beachten ist bei diesem Ansatz, dass die direkte Kontrolle am Satelliten immer noch von der Kommandoanwendung ausgeht. Wie es im Falle der Referenzszenarien vorgesehen ist, wird daher zum Beispiel die Anwendung zur Lageregelung des Satelliten (*Attitude Control Application*) direkt von der Kommandoanwendung - und nicht vom APS - angesteuert.

Damit wären die Interaktionen mit den vom APS unmittelbar betroffenen Anwendungen erläutert. Mittelbar betroffene Teilnehmer befinden sich vor allem auf dem Bodensegment. Der Operator als Anwender bedient sich zur Interaktion mit dem Satelliten in der Regel zweier Anwendungen, einer zur Kommandierung des Satelliten (*Command Ground Application*) und einer zur Erfassung der Telemetrie des Satelliten (*Telemetry Ground Application*). Diese Anwendungen generieren und verarbeiten Kommandos (*Cmd*) und Housekeeping-Daten (*Hk*). Bei der Verwendung des APSs müssen die Anwendungen entsprechend angepasst werden. Der Transfer der Daten (Kommandos und Housekeeping) zwischen den beiden Raumfahrtsegmenten per Funkstrecke wird mit Hilfe spezieller Kommunikationsanwendungen (*COM Application* und *COM Ground Application*) koordiniert. Diese implementieren speziell dafür vorgesehene Übertragungsprotokolle, und übersetzen TCs in Kommandos (Cmd) und TMs in Housekeeping-Daten (Hk). Hierbei sollte beachtet werden, dass es sich bei den Anwendungen auf dem Raumsegment um RODOS-Anwendungen handelt. In Folge dessen wandelt die Kommunikationsanwendung die einzelnen Datenpakete vom Bodensegment auch in Botschaften der RODOS-Middleware um. Dieser Schritt ist erforderlich, da die Anwendungen des Bodensegments nicht notwendigerweise kompatibel zu RODOS sind.

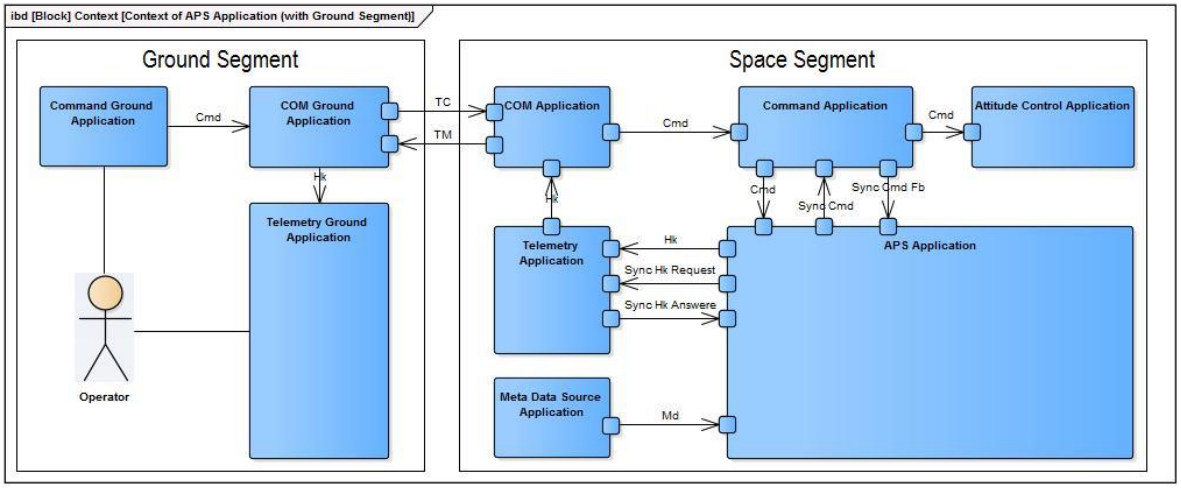


Abbildung 7-14: Systemkontext des APS

Für die weitere Beschreibung des APS erscheint es sinnvoll, die Details für den Datenverkehr zwischen den Segmenten auszublenden und sich stattdessen ausschließlich mit dem Raumsegment zu befassen. Die Abläufe zur Realisierung der autonomen Funktionalitäten finden fast ausschließlich dort statt. Aus diesem Grund wurde der oben eingeführte Kontext abstrahiert (siehe Abbildung 7-15), indem die neu eingeführte Ground-Anwendung die Aspekte der Funkkommunikation sowie die Steuerung durch den Operator kapselt.

Im Abschnitt 7.2.6.3 wird die indirekte Kommunikation zwischen dem APS und der Bodenstation als besonderer Aspekt wieder aufgegriffen.

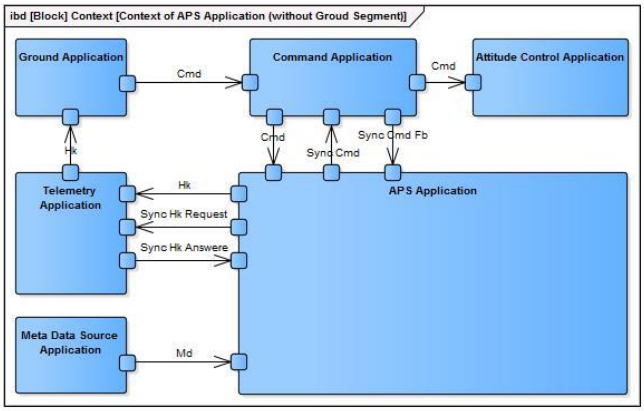


Abbildung 7-15: abstrahierter Systemkontext des APS mit der Ground-Anwendung

7.2.5 Systemsynthese

Der nächste Schritt im Entwurfsprozess ist die Synthese des APSs. Zunächst wird dessen funktionales Verhalten (siehe Abbildung 7-16), ausgehend von seinem Systemkontext und den an ihn gestellten Anforderungen, spezifiziert. In einem weiteren Schritt erfolgt anschließend die Umsetzung.

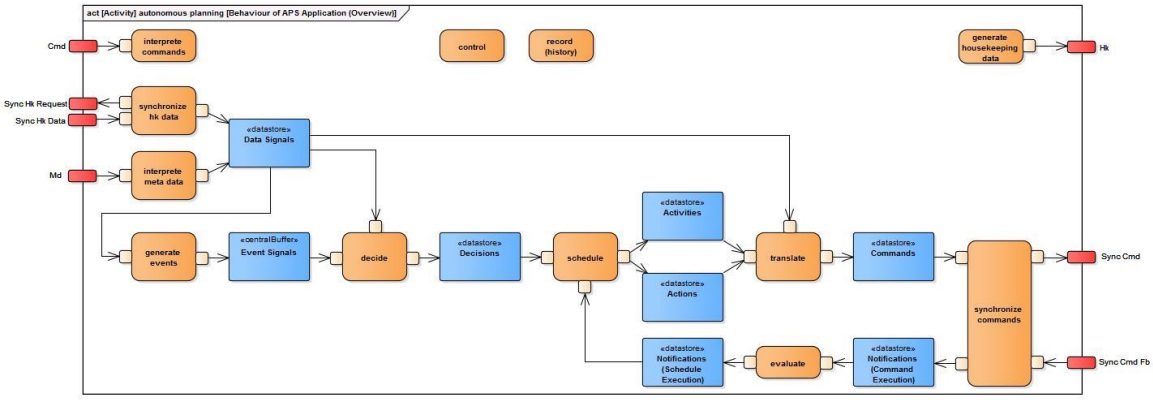


Abbildung 7-16: funktionales Verhalten der autonomen Planung (Übersicht)

Die Parameter im Aktivitäten-Diagramm entsprechen den einzelnen RODOS-Topics, mit denen die APS Anwendung im Kontext eingebettet ist. Aus Gründen der Übersichtlichkeit sind in der Abbildung nur Datenflüsse dargestellt, die unmittelbare Bestandteile des eigentlichen autonomen Planungsprozess sind.

Die autonome Planung besteht im Wesentlichen aus den nachfolgend gelisteten Funktionalitäten, die im Anschluss kurz erläutert werden:

- Steuerung (*control*)
- Kommandierung (*interprete commands*)
- Zustandsdatenerfassung (generate *housekeeping data*)
- Zustandsdatensynchronisation (*synchronize hk data*)
- Meta-Daten-Interpretation (*interprete meta data*)
- Ereignisgenerierung (*generate events*)
- Entscheidungsfindung (*decide*)
- Ablaufplanung (*schedule*)
- Planübersetzung (*translate*)
- Kommandosynchronisation (synchronize commands)

- Planevaluierung (*evaluate*)
- Aufzeichnung (*record/history*)

Das APS ist zuerst wie eine gewöhnliche Software-Anwendung in der Raumfahrt zu betrachten und besitzt daher drei Standardfunktionalitäten: Erstens, eine interne Steuerung (*control*), deren Aufgabe darin besteht, sämtliche Kontroll- und Datenflüsse der Anwendung zu koordinieren. Zweitens, einen Mechanismus zur eigenen Kommandierung (*interprete commands*) und drittens, einen zur Erfassung der eigenen Zustandsdaten (*generate housekeeping data*).

Auf diesem Grundgerüst werden die Funktionalitäten zur Umsetzung der autonomen Planung aufgesetzt. Diese kann als ein Standardprozess, mit einer Eingabe, einer Verarbeitung und einer Ausgabe aufgefasst werden. Zusätzlich gibt es zur Kontrolle auch eine Rückführung von der Ausgabe auf die Verarbeitung in Form von Rückmeldungen (*notifications*).

Eingangsseitig werden Daten (*Data Signals*) aufbereitet und zwischengespeichert. Sie werden als Grundlage für die nachfolgende Verarbeitung genutzt. Als Quelle dient zum einen die Anwendung zur Zustandsdatenverwaltung (siehe Kontext) und zum anderen potentiell jede Anwendung, die Meta-Daten erzeugen und ausgeben kann. Es ist bekannt, dass die Zustandsdatenverwaltung eine eigene Liste mit den Daten führt. Diese können mittels Synchronisation (*synchronize hk data*) vom APS aus wahlweise abgeglichen werden. Die Daten werden über das Topic *sync hk data* eingelesen. Sie werden von den Quellen entweder unaufgefordert bereitgestellt oder müssen im Rahmen der Synchronisation zuerst erfragt (Topic: *sync hk request*) werden.

Ein weiterer wichtiger Schritt im Aufbereiten der Eingabe für die autonome Planung besteht im Generieren von Ereignissen (*generate events*). Diese können entweder direkt aus den zuvor erwähnten Eingangsdaten extrahiert werden oder einen komplett anderen Ursprung (z.B. Zeit-Timer) haben.

Nach der Eingabe (Daten und Ereignisse) erfolgt die Verarbeitung. Diese besteht im Wesentlichen aus drei Hauptfunktionalitäten: Zuerst müssen auf Basis der Eingabe Entscheidungen (*decissions*) getroffen werden (*decide*). Diese geben das durch die autonome Planung gewünschte Verhalten des Satelliten vor. Zum Zeitpunkt der Erstellung sind die Entscheidungen aber noch nicht umgesetzt. Daher müssen diese im nächsten Schritt in die Ablaufplanung (*schedule*) mit einbezogen werden. Hauptgegenstand der Ablaufplanung sind die Aktivitäten (*Activities*) und die Aktionen (*Actions*), die zusammengefasst den Ablaufplan ergeben. Aktivitäten erfüllen, einzeln betrachtet, jeweils eine bestimmte Aufgabe (z.B. Batterie aufladen, siehe Referenzszenarien). Eine Aktivität kann im Laufe ihres Lebenszyklus verschiedene Zustände (z.B. Laufend, Angehalten, Abgeschlossen) annehmen. Deren Übergänge, die im Rahmen der Ablaufplanung zeitlich festgelegt werden, werden durch Aktionen (z.B. Start, Pause, Stop) eingeleitet. Jede Aktion hat im Regelfall daher eine zugewiesene Aktivität. Hiervon gibt es eine Ausnahme: Durch die Entscheidungsfindung werden einzelne isolierte Aktionen direkt beeinflusst. Dies ist jedoch nur dann erlaubt, wenn die betroffenen Aktionen keine zugewiesene Aktivität haben. Entscheidungsfindung und Ablaufplanung werden zusammengefasst auch als Planung bezeichnet.

Damit einzelne Aktionen vom Satelliten ausgeführt werden können, müssen sie in passende Kommandosequenzen (*Commands*) übersetzt (*translate*) werden. Im Anschluss können diese abgefertigt, d.h. zum Bordcomputer übertragen, und dort aufgerufen werden. Da der Bordcomputer bzw. die in ihm enthaltende Anwendung zur Kommandoverwaltung über eine eigene auszuführende Kommandoliste verfügt, muss eine zuvor notwendige Synchronisation (*synchronize commands*) erfolgen. Diese kann, wenn gewünscht, kommandobezogene Rückmeldungen (*notifications/command execution*) von der Kommandoverwaltungsanwendung evaluieren (*evaluate*), und der Ablaufplanung selbst Rückmeldungen (*notifications/schedule execution*) geben. Übersetzung, Synchronisation und Evaluierung werden zusammengefasst auch als Planausführung bezeichnet.

Als letztes wird die Funktionalität zur Aufzeichnung (*record history*) der im autonomen Planungsprozess erfolgten Abläufe erläutert. Die von ihr hinterlegten Daten ermöglichen es dem Operator, das Verhalten des APS nachzuvollziehen. Sämtliche relevanten Vorkommnisse im System müssen chronologisch erfasst und mit Zeitstempeln versehen werden.

Wegen der hohen Komplexität der bisher beschriebenen Funktionalitäten müssen weitere untergeordnete Komponenten im Rahmen der Synthese erschaffen werden, die, durch ihre Interaktionen untereinander, jeweils Teile der geforderten Funktionalität implementieren. Das APS hat daher die in Abbildung 7-17 dargestellte innere Struktur.

Das APS besteht generell aus aktiven und passiven Komponenten. Als aktiv werden jene bezeichnet, die einen verarbeitenden Charakter haben und die zuvor erwähnten Funktionalitäten implementieren bzw. kapseln. Sie werden vom Kontrollfluss der Anwendung entweder direkt oder indirekt aufgerufen, und verändern andere Komponenten. Passive Komponenten hingegen sind lediglich strukturierte Daten ohne verarbeitenden Charakter.

Zu den wichtigsten aktiven Komponenten gehören:

- Hauptsteuerung (*Control*)
- Kommandoverwaltung (*Command Manager*)
- Telemetrieverwaltung (*Telemetry Manager*)
- Datenverwaltung (*Data Manager*)
- Ereignisverwaltung (*Event Manager*)
- Entscheider (*Decider*)
- Ablaufplaner (*Scheduler*)
- Planausführer (*Executor*)
- Datenrekorder (*History Manager*)

Allen aktiven Komponenten ist gemein, dass sie in der Regel direkt nur mit passiven Komponenten interagieren. In der Abbildung ist dieser Bereich als Datenpool (*Data Pool*) gekennzeichnet. Hierbei handelt es sich um eine Kapselung. Dahinter verbergen sich mehrere, in Beziehung zueinander stehende, Datenstrukturen. Sie werden im Laufe der folgenden Abschnitte bei der Beschreibung der verschiedenen Interaktionen im Detail betrachtet.

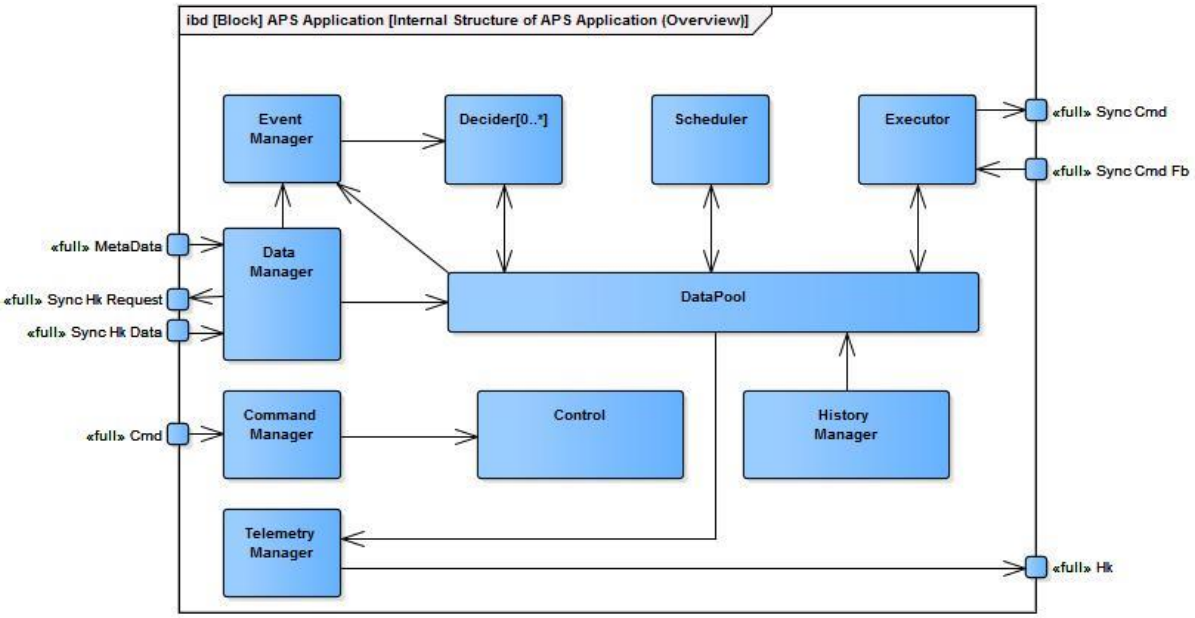


Abbildung 7-17: interne Struktur des APS

7.2.5.1 Hauptsteuerung

Die primäre Aufgabe der Hauptsteuerung ist die Sicherstellung des Kontrollflusses (d.h. der Programmabläufe) in der Anwendung. Sämtliche Interaktionen werden von ihr koordiniert. Der Ablauf ist in der folgenden Abbildung 7-18 dargestellt.

Zuerst werden die Anwendung und seine Komponenten mit Standardwerten initialisiert (*initialize*). Danach wird eine endlose Schleife gestartet, in der zyklisch eine in der Hauptsteuerung enthaltene Zustandsmaschine (siehe Abbildung 7-19) angesteuert wird. Diese ermöglicht einen geordneten Betrieb des autonomen Planungssystems. Im Anschluss an die Zustandsmaschine werden mehrere Hauptkomponenten der Anwendung angesteuert. Dazu gehören die Kommandoverwaltung (*manage commands*), die Telemetrieverwaltung (*manage telemetry*), die Datenverwaltung (*manage data*) und die Eventverwaltung (*manage events*). Damit endet der Schleifenkörper, sodass der Kontrollfluss in einem neuen Zyklus wieder mit der Zustandsmaschine beginnen kann. In der Schleife werden jedoch nicht alle Komponenten, die für den autonomen Planungsprozess entscheidend sind, berücksichtigt. Die für die autonome Planung wichtigsten Komponenten, dazu gehören die Entscheider, der Ablaufplaner sowie der Planausführer, laufen in jeweils separaten RODOS-Threads. Diese Maßnahme dient dazu, um auch bei längeren Berechnungen eine effektive Ansteuerung des Satelliten (durch den Planausführer) zu gewährleisten und gegenseitige Blockierungen der Komponenten untereinander zu verhindern. Die involvierten RODOS-Threads werden bei bestimmten Übergängen der Zustandsmaschine gestartet bzw. gestoppt.

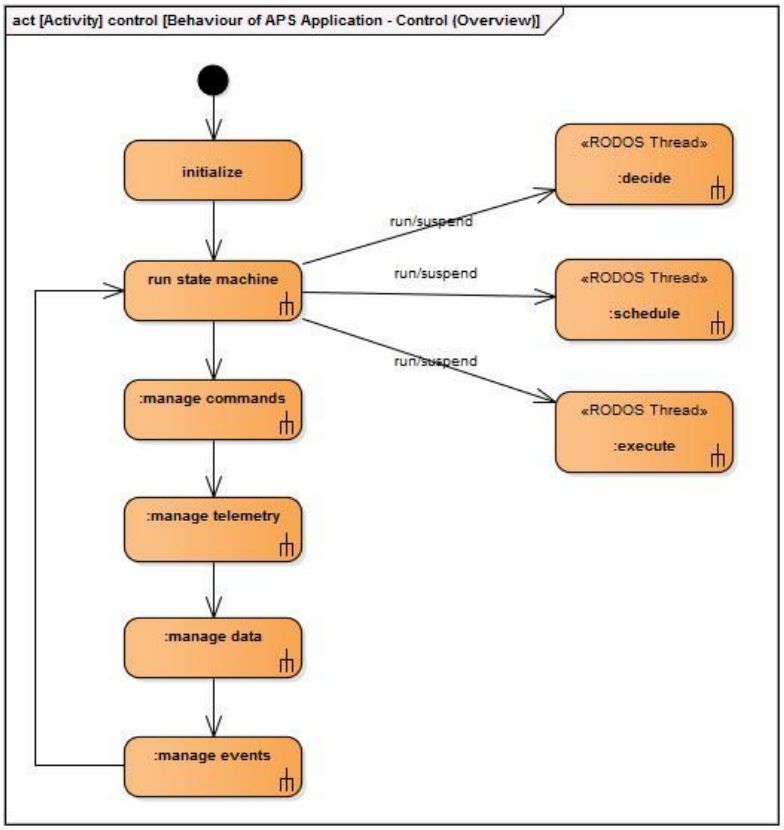


Abbildung 7-18: Abläufe in der Hauptsteuerung

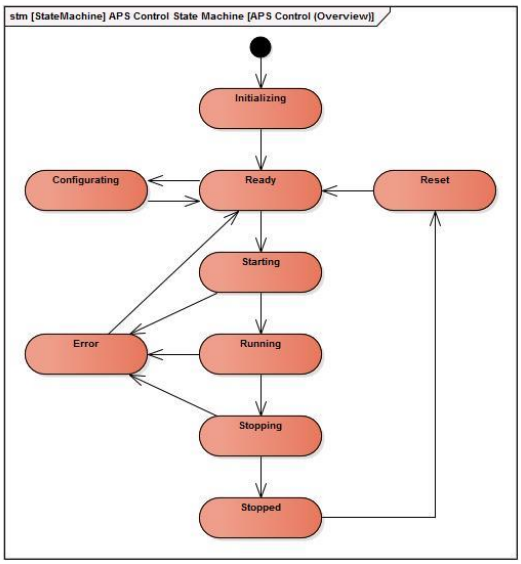


Abbildung 7-19: Zustände der Hauptsteuerung

In der Zustandsmaschine der Hauptsteuerung werden Übergänge entweder automatisch oder gezielt durch Kommandierung ausgelöst. Zu Beginn startet die Hauptsteuerung mit einer Initialisierung im Zustand *Initializing* und wechselt im Anschluss in *Ready,* um ihre Bereitschaft zu signalisieren. Als nächstes muss das autonome Verhalten des APSs eingestellt werden. Dazu ist ein Übergang in den Zustand *Configurating* notwendig. Nur in diesem lässt sich das gewünschte Verhalten, d.h. die Definition, des autonomen Planungssystems für eine bestimmte Satellitenmission einstellen. Dieser Vorgang wird allgemein als Konfiguration bezeichnet. Zum Beispiel müssen die Aktivitäten und Ressourcen des Satelliten sowie deren Beziehungen zueinander definiert werden. Ebenso müssen Satellitenkommandos eingepflegt werden. Alternativ können die Einstellungen aus einem nichtflüchtigen Speicher geladen und verwendet werden, sofern die Plattform mit solch einem ausgestattet ist. Wenn alle Informationen vom Operator eingegeben wurden, muss das System wieder zurück in den Ausgangszustand *Ready* versetzt werden. Damit sind die Vorbereitungen für den autonomen Betrieb abgeschlossen. Dieser kann dann mit einem passenden Kommando vom Operator eingeleitet werden. Nach einer kurzen Vorbereitung (*Starting*) ist das System einsatzbereit (*Running*) und greift autonom in den Betrieb des Satelliten ein. Hierbei gilt es zu berücksichtigen, dass es, wie in den Anforderungen gefordert, einen aktiven und einen passiven Modus gibt. Nur im aktiven Modus wird der Satellit durch das APS mittels Kommandosynchronisation beeinflusst. Beim passiven Modus werden zwar Pläne vom Planungssystem autonom erstellt, jedoch nicht zur Ausführung an den Satelliten weitergegeben. Der autonome Betrieb im Zustand *Running* wird so lange aufrechterhalten, bis die Beendigung mit einem passenden Kommando eingeleitet wird. Wie beim Starten ist auch dieser Vorgang zweistufig. Nach einer kurzen Vorbereitung (*Stopping*) ist das System angehalten (*Stopped*) und greift nun nicht mehr autonom in den Betrieb des Satelliten ein. Der Operator kann jedoch noch diverse Informationen (über den Datenrekorder) zu den letzten Tätigkeiten des APS einholen. Um wieder die Ausgangslage (*Ready*) zu erreichen, muss das System zurückgesetzt werden (*Reset*). Dies ist notwendig, um alle Daten der letzten autonomen Planung (z.B. Ereignisse und Verläufe von Ressourcen) zu löschen, damit diese nicht im Widerspruch zu neuen Planungen stehen. Die im Rahmen der Konfiguration (*Configurating*) eingespeisten Daten sind vom Löschvorgang ausgeschlossen. Die bisher genannten Zustandsübergänge entsprechen dem normalen Verlauf. Während des autonomen Betriebes können aber auch schwerwiegende Fehler auftreten. In diesem Fall ist eine gesonderte Fehlerbehandlung notwendig: Die Zustandsmaschine springt in einen speziell dafür vorgesehenen Fehlerzustand (*Error*) über. Dieser kann dazu verwendet werden, bestimmte Fehlerspeicher auszulesen und auszuwerten. In der Regel lassen sich Fehler auf falsche Einstellungen während der Konfiguration zurückführen. Im Anschluss ist ein Wechsel in den Ausgangszustand *Ready* vorgesehen.

7.2.5.2 Datenverwaltung

Die Datenverwaltung (*Data Manager*) des autonomen Planungssystems hat die Aufgabe, Eingabedaten bzw. Stimuli für die nachfolgenden aktiven Komponenten (z.B. Entscheidungsfindung, Ablaufplanung, Planausführung) in passender Form bereitzustellen. Zu diesem Zweck sind Vorarbeiten notwendig, da sie die benötigten Daten aus den über die Topics *Hk Sync Data* und *Meta Data* versendeten Botschaften extrahieren muss. Die gewonnenen Daten müssen anschließend im Daten-Pool an den ihnen zugewiesenen Stellen zwischengespeichert werden (siehe Abbildung 7-20).

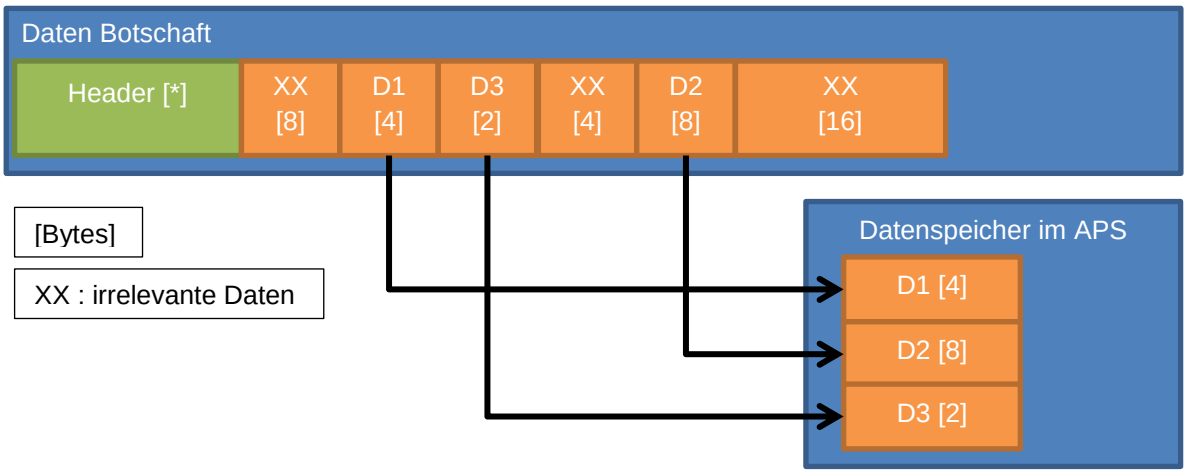


Abbildung 7-20: Extrahierung ausgewählter Daten

Im Datenpool wird vom Datenmanager ein Abbild aller für die autonome Planung relevanten Daten verwaltet. Es gibt zwei verschiedene Arten von Datenquellen: Zum einem die Telemetrie-Anwendung, die Zustandsdaten (Housekeeping) des Satelliten verwaltet, und zum anderem Anwendungen, die die sogenannten Meta-Daten publizieren. Dies sind Daten, deren Semantik bekannt ist und die beschreibenden Charakter haben. Dies ist eine Grundvoraussetzung, damit die Daten wie gefordert zugeordnet werden können. Beide Datenquellen befinden sich im Kontext der APS Anwendung. Wenn sie neue Botschaften verschicken, werden die betroffenen Daten im Daten-Pool mit den neuen extrahierten Werten aktualisiert.

Manche Anwendungen im Kontext vom APS senden ihre Daten unaufgefordert, andere müssen im Rahmen der Synchronisation dazu aufgefordert werden. Im letzten Fall muss die Datenverwaltung über das Topic *Sync Hk Request* dementsprechende Anfragen verschicken.

Im Folgenden soll nun die Interaktionen der Datenverwaltung genauer erläutert werden. Der Ablauf ist in der nachfolgenden Abbildung 7-21 dargestellt.

Die Datenverwaltung wird durch die Hauptsteuerung des autonomen Planungssystems zyklisch aufgerufen. Zu Beginn wird sie überprüfen, ob ggfs. im Rahmen der Synchronisation Datenanfragen an die Telemetrie-Anwendung gesendet werden müssen. Sollte dies der Fall sein, werden entsprechende Anfragen erstellt und über das Topic *Sync Hk Request* versendet. In der Regel werden Anfragen zyklisch fällig. Die Zeiten unterscheiden sich je nach Szenario. Im nächsten Schritt verarbeitet die Datenverwaltung empfangene Botschaften. Diese sind je nach Quellanwendung über die Topics *Sync Hk Data* oder *Meta Data* eingegangen. Generell wird jede erhaltene Botschaft zwischengespeichert und schließlich ausgewertet. Hierfür sind zusätzliche Informationen erforderlich. Diese werden im Rahmen der Konfiguration vorgegeben. Besonders festzuhalten sind die Art der Daten und die Zuordnungen zu den einzelnen Botschaften. Bei einer erfolgreichen Extraktion werden die gewonnenen Daten im Daten-Pool abgelegt. An dieser Stelle können die Daten rein technisch betrachtet auch als Datensignale aufgefasst werden, da sie Träger einer Information sind. Im letzten

Schritt wird noch der Ereignismanager über den Empfang der Botschaft und deren Auswertung informiert, da diese Botschaft möglicherweise ein Ereignis auslöst (siehe Abschnitt 7.2.5.3). Damit endet die Abfolge der Datenverwaltung, sodass der Kontrollfluss wieder zurück an die zentrale Steuerung des autonomen Planungssystems wechselt.

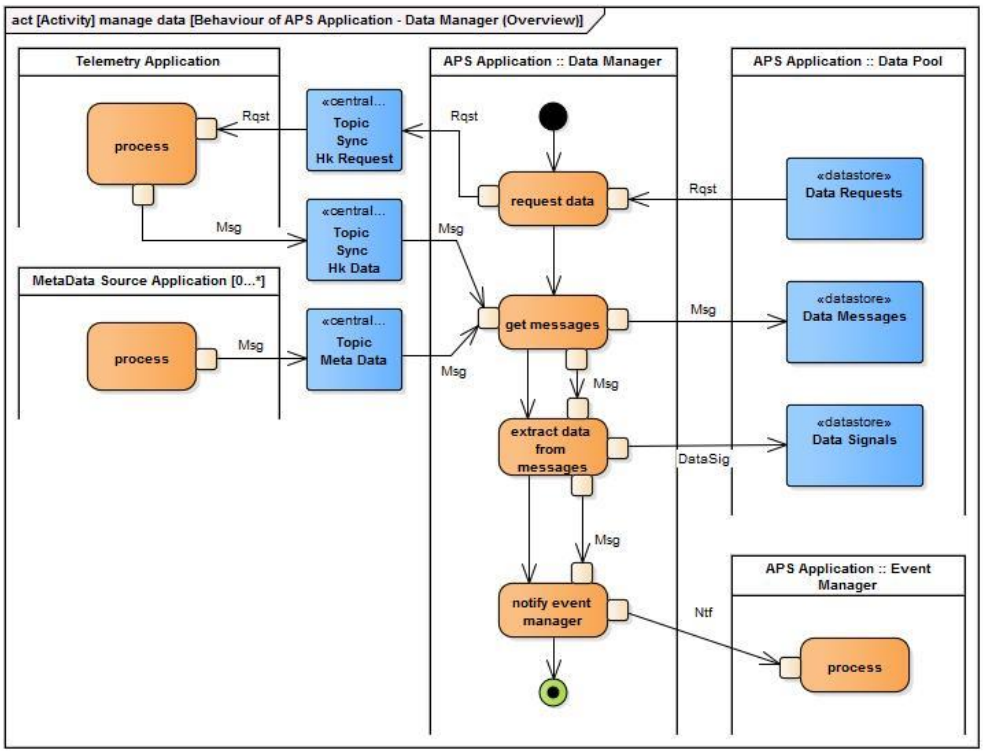


Abbildung 7-21: Interaktionen zur Datenverwaltung beim autonomen Planungssystem

Wie bereits oben erwähnt wurde, wird beim Extrahieren detailliertes Wissen über die Struktur und die Zusammensetzung der beteiligten Botschaften benötigt. Diese Informationen müssen vom Operator bei der Konfiguration des autonomen Planungssystems in Form von Definitionen bereitgestellt werden. Die nachfolgende Abbildung 7-22 zeigt die statischen Zusammenhänge.

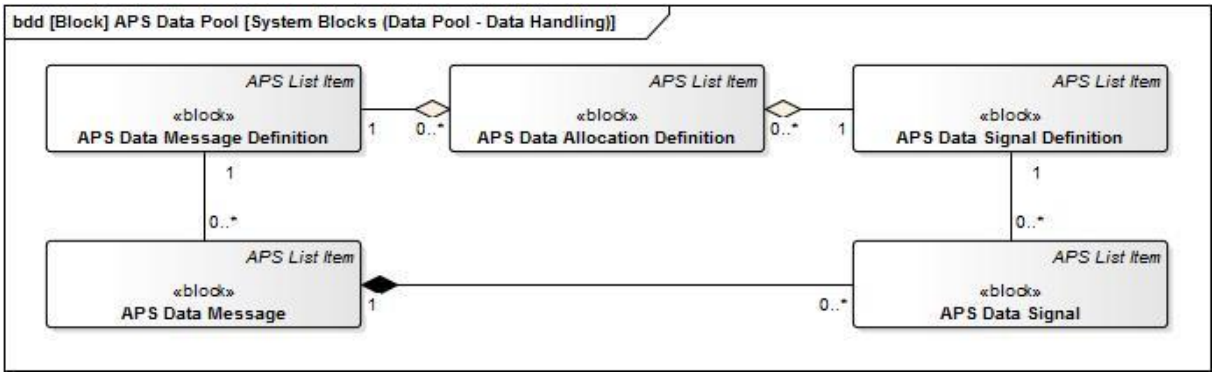


Abbildung 7-22: Passive Komponenten der Datenverwaltung im Daten-Pool

Die Datenverwaltung führt im Datenpool drei Listen, um die verschiedenen Arten von Botschaften (*APS Data Message Definition*) und Datensignalen (*APS Data Signal Definition*), sowie deren Zusammenhänge (*APS Data Allocation Definition*) untereinander, zu definieren. Bei der Auswertung konkreter Botschaften (*APS Data Message*) wird auf die Definition zugegriffen, um die enthaltenen Datensignale (*APS Data Signal*) zu extrahieren.

7.2.5.3 Ereignisverwaltung

Ereignisse sind ein Hilfsmittel der autonomen Planung und werden wie normale Datenbotschaften mit besonderem Informationsgehalt behandelt. Die Aufgabe der Ereignisverwaltung besteht darin, Ereignisse zu erzeugen und im Anschluss je eine Kopie davon an alle Entscheider zu schicken. Deren Entscheidungsabläufe können auf den eingegangen Ereignissen basieren. Der gesamte Vorgang der Erzeugung wird mit Hilfe von Regeln, die aus einem Auslöser (Trigger), optional aus einer Bedingung (Condition) und einem ausführenden Element bestehen, koordiniert.

Im Folgenden soll die Interaktion der Ereignisverwaltung genauer erläutert werden. Der Ablauf ist in der nachfolgenden Abbildung 7-23 dargestellt.

Die regelbasierte Ereignisverwaltung kennt zwei verschiedene Typen von Auslösern: Zum einem sind das die zeitbezogenen Auslöser. Die Quelle ist hierbei eine interne Uhr, die abhängig von ihrer Konfiguration (einmalig oder zyklisch) zu bestimmten Zeitpunkten einen oder mehrere Alarme auslöst. Zum anderen sind das die datenbasierten Auslöser in Form von Datenbotschaften. Die Quelle ist hierbei die im vorherigen Abschnitt vorgestellte Datenverwaltung. Wie bereits erwähnt wurde, übermittelt die Datenverwaltung der Ereignisverwaltung jede einzeln verarbeitete Datenbotschaft (*Data Message)*. Diese kann von der Ereignisverwaltung dahingehend überprüft werden, ob der entsprechenden *Data Message Definition* ein Auslöser zugeordnet wurde. Generell wird, je nach Typ des Auslösers, mit dem Eintreten des Ereignisses auch eine bestimmte Menge an Parametern festgehalten. Diese fließen in die weitere Verarbeitung ein. Beispiele hierfür sind der Zeitpunkt des Alarms, oder die Datensignale einer eingetroffenen Datenbotschaft.

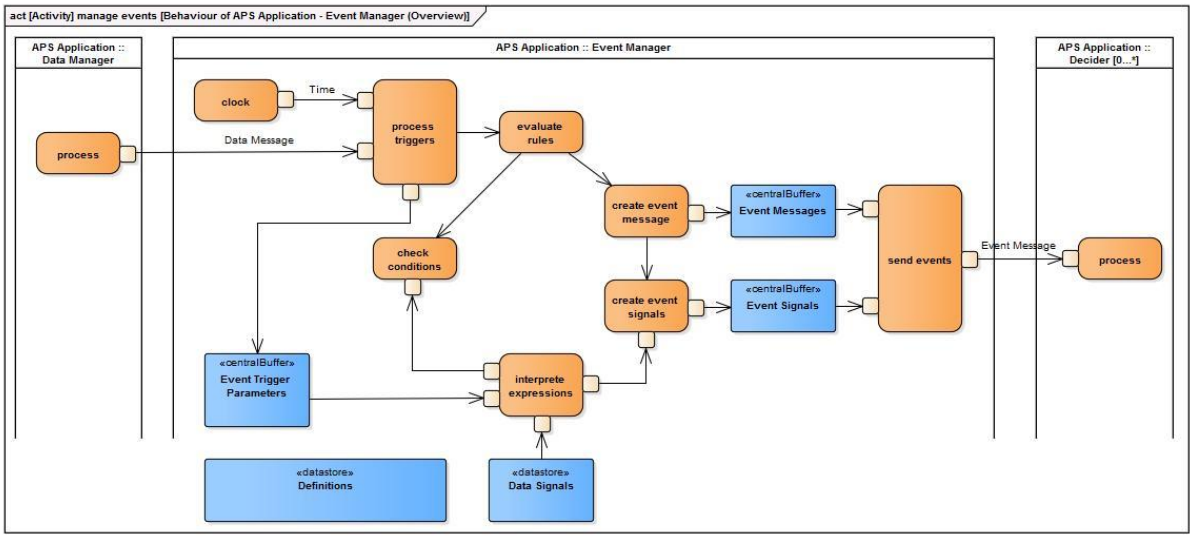


Abbildung 7-23: Interaktionen zur Ereignisverwaltung beim autonomen Planungssystem

Die Auslöser werden permanent überwacht und stoßen bei ihrer Aktivierung die ihnen zugewiesenen Regeln an. Als Folge wird für jede angestoßene Regel überprüft, ob deren Bedingung erfüllt ist. Aus logischer Sicht handelt es sich hierbei um boolsche Aussagen. Teilausdrücke der Aussagen können auf bestimmte Datenbereiche (z.B. Datensignale) im Datenpool oder auf die oben erwähnten Parameter der Auslöser verweisen und lassen sich zu komplexeren Ausdrücken kombinieren. Sollte die Bedingung erfüllt sein, wird die Regel ausgeführt. Dabei wird das entsprechende Ereignis mitsamt seinen Signalen als Botschaft erstellt und anschließend an alle Entscheider geschickt. Bei der Signalerstellung für die entsprechenden Ereignisse werden ebenfalls Ausdrücke ausgewertet.

Der oben beschriebene Vorgang benötigt detailliertes Wissen über die Struktur und die Zusammensetzung der beteiligten Botschaften und Ereignisse. Diese Informationen müssen vom Operator bei der Konfiguration des autonomen Planungssystems bereitgestellt und im Datenpool der Anwendung abgelegt werden. Die Informationen sind wie in Abbildung 7-24 dargestellt strukturiert.

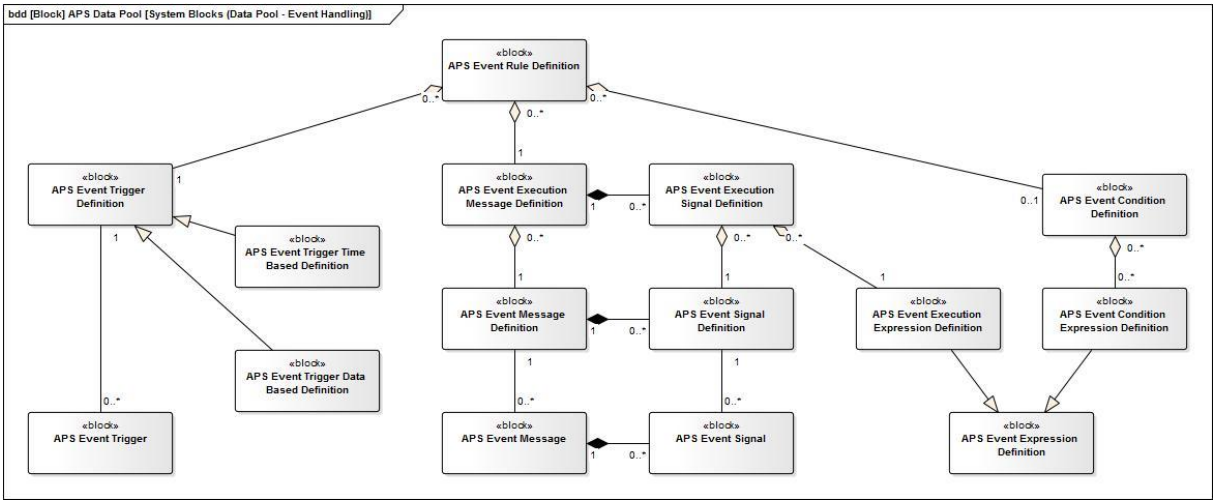


Abbildung 7-24: Blöcke des Datenpools für die Eventverwaltung

Hauptbestandteil der Ereignisverwaltung sind die Regeldefinitionen (*APS Event Rule Definition*). Jeder Regeldefinition sind ein Auslöser (*APS Event Trigger Definition*), eine Bedingung (*APS Event Condition Definition*) und ein ausführender Teil zur Ereigniserstellung (*APS Event Execution Message Definition*) zugewiesen. Auslöser, Bedingungen und Ausführung können aus Gründen der Wiederverwendbarkeit in mehreren Regeln gleichzeitig verwendet werden. Es gibt zwei verschiedene Typen von Triggern (*APS Event Trigger Definition*), den zeitbasierten Trigger (*APS Event Trigger Time Based Definition*) und den datenbasierten Trigger (*APS Event Trigger Data Based Definition*). Jeder tatsächlich eingetretene Trigger (*APS Event Trigger*) hat eine Referenz auf seine entsprechende Definition. Bei der Erstellung von Ereignissen (*APS Event Message*) und deren Signalen (*APS Event Signal*) kommen die Blöcke *„APS Event Execution Message Definition“* und *„APS Event Execution Signal Definition“* zum Einsatz. Sie beschreiben genau, wie die entsprechenden Ereignisbotschaften aufzubauen sind. Im APS zulässige Ereignisdefinitionen werden mit Hilfe der Blöcke „*APS Event Message Definition*“ und „*APS Event Signal Definition*“ beschrieben und bekanntgegeben. Zur Definition der Ereignissignale und der Regelbedingungen werden entsprechende logische Ausdrücke (*APS Event Execution Expression Definition* und *APS Event Condition Expression Definition*) benötigt, die gemeinsam von „*APS Event Expression Definition*“ abgeleitet sind. Ausdrücke werden vom Operator vorgeben und zu Beginn des autonomen Betriebes in entsprechenden Code interpretiert.

7.2.5.4 Entscheidungsfindung

Die Entscheidungsfindung ist ein Prozess der autonomen Planung. Sie dient der Verarbeitung von Eingangsinformationen (Daten- und Ereignissignale) und läuft in einem separaten RODOS-Thread. Das Ziel besteht darin, Anweisungen bzgl. der für den Satelliten zu planenden Aktivitäten und Aktionen zu generieren. Die nachfolgende Abbildung 7-25 zeigt die Funktionsweise in stark vereinfachter Form.

Die wichtigsten an der Entscheidungsfindung beteiligten Komponenten sind die Entscheider (*Decider*). Deren Verhaltensweisen können im Rahmen der Konfiguration angepasst werden. Die Aufgabe der Entscheider besteht darin, entweder neue Aktivitäten und/oder Aktionen für den Satelliten zu erstellen, bereits bestehende anzupassen, oder abzubrechen. Da die zu planenden Aktivitäten und Aktionen generell direkt nur von der nachfolgenden Ablaufplanung verwaltet werden, können Entscheider lediglich entsprechende Anweisungen in Form von Entscheidungen (*Decisions*) geben. Diese müssen anschließend von der Ablaufplanung umgesetzt werden.

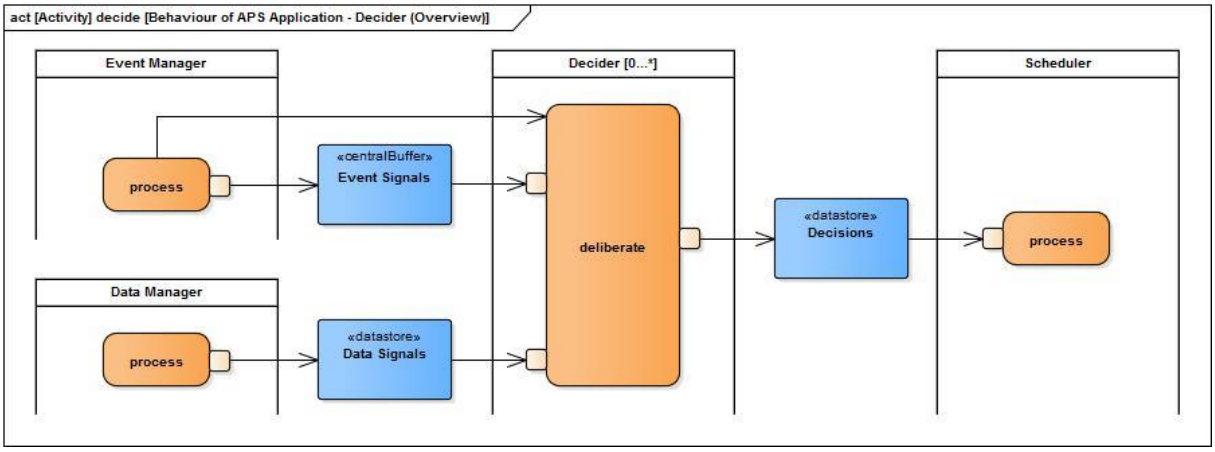


Abbildung 7-25: Interaktionen zur Entscheidungsfindung (Übersicht)

Das autonome Planungssystem kann so konfiguriert werden, dass nicht nur ein, sondern zwei oder sogar mehrere Entscheider zur selben Zeit koexistieren. Der in einem separaten RODOS-Thread laufende Entscheidungsprozess wird im Regelfall jeden einzelnen Entscheider zyklisch zur Arbeit anstoßen. Die Berechnungen zur Entscheidungsfindung basieren dabei eingangsseitig ausschließlich auf Daten- und Ereignissignalen (siehe Abschnitte 7.2.5.2 und 7.2.5.3). Außerplanmäßig können in besonderen (zeitkritischen) Fällen die Entscheider aber auch direkt durch die Ereignisverwaltung zur Behandlung eines Ereignisses aufgerufen werden.

Wie bereits erwähnt wurde, ist das autonome Planungssystem so konzipiert, dass mehrere Instanzen an Entscheidern zur selben Zeit aktiv sein können. Diese können sich nicht nur hinsichtlich ihrer vom Operator konfigurierten Einstellungen, sondern auch grundsätzlich in Bezug auf den Aufbau voneinander unterscheiden. Sie müssen lediglich dieselben Grundfunktionen (d.h. die Erstellung von Anweisungen) beherrschen. Dies hat zum Vorteil, dass zum einen die Entscheider verschiedenen Paradigmen (z.B. ereignisbasiert, zielbasiert) folgen können und dass zum anderen das autonome Planungssystem zukünftigen Erweiterungen (auch programmtechnischen) offensteht.

Die Umsetzung zur Erreichung der oben genannten Vorteile erfolgt wie in Abbildung 7-26 dargestellt.

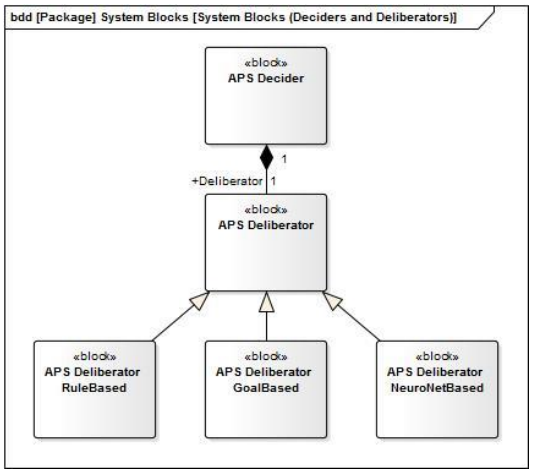


Abbildung 7-26: Beziehung zwischen Entscheider und Prüfer

Jeder Entscheider verfügt über eine untergeordnete, als Prüfer (*APS Deliberator*) bezeichnete Komponente, in der das Wissen und die Intelligenz, um autonome Entscheidungen zu treffen, gekapselt sind. Vom technischen Standpunkt aus betrachtet wird Abstraktion angewandt. Es gibt mehrere abgeleitete Typen von Prüfern, die alle vom Basistyp *APS Deliberator* erben. Auf diese Weise lassen sich verschiedene Prinzipien der Entscheidungsfindung, die jeweils verschiedenen Paradigmen folgen, umsetzen:

- regelbasierte Entscheidungsfindung (*APS Deliberator RuleBased*)
- zielbasierte Entscheidungsfindung (*APS Deliberator GoalBased*) (*)
- neuronal-netzwerkbasierte Entscheidungsfindung (*APS Deliberator NeuroNetBased*) (*)

(*) Diese Arten der Entscheidungsfindung werden in der vorliegenden Dissertation nicht weiter untersucht. Sie sind jedoch als Schwerpunkte weiterführender Vorhaben (siehe Kapitel 10 Ausblick), die sich in der Vorbereitungs- und Antragsphase befinden, vorgesehen.

Regelbasierte Entscheidungsfindung

Für diese Art der Entscheidungsfindung ist der regelbasierte Prüfer (*APS Deliberator RuleBased*) zuständig. Sein struktureller Aufbau ist in Abbildung 7-27 dargestellt. Er trifft seine Entscheidungen anhand zuvor festgelegter Regeln (*APS Rule*). Diese werden im Rahmen der Konfiguration vorgegeben und in einer speziellen Liste (*APS Rule List*) verwaltet. Eine Regel besteht im Wesentlichen aus drei Bestandteilen:

- den Auslöser (*APS Trigger*)
- die Bedingung (*APS Condition*)
- die Ausführung (*APS Execution*)

Eine Regel wird vom Prüfer nur dann ausgewertet, wenn sie durch eine bestimmte Ursache ausgelöst wurde. Diese kann entweder in der Meldung eines Ereignisses durch die Ereignis-Verwaltung oder im Verstreichen einer bestimmten Zeitspanne liegen. Zum Zweck der Auswertung stehen zwei konkrete Auslöser, der *APS Trigger Event* und der *APS Trigger Time*, zur Verfügung. Beide sind von einem Basis-Trigger (*APS Trigger*) abgeleitet.

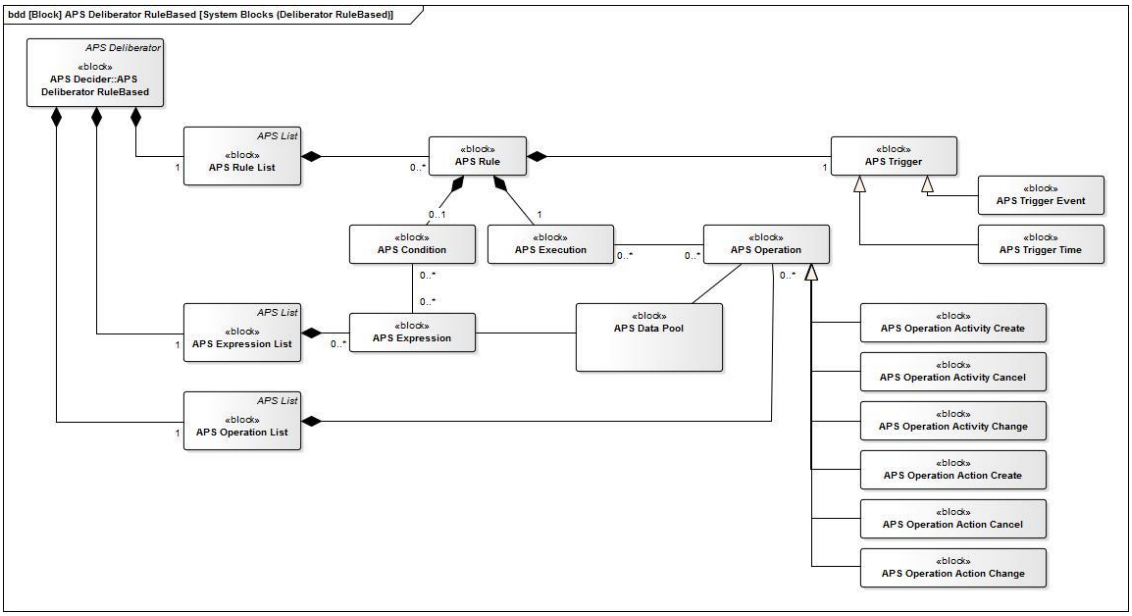


Abbildung 7-27: Blöcke des Prüfers *APS Deliberator RuleBased*

Im Falle einer Auslösung muss der Prüfer feststellen, welche Regeln mit ihr verknüpft sind. Diese gilt es dann hinsichtlich ihrer Bedingungen (*APS Condition*) zu überprüfen. Ihre Erfüllung muss gegeben sein, damit die entsprechenden Regeln ausgeführt werden können. Dabei muss auch der Fall berücksichtigt werden, dass keine Bedingung vorliegen kann, da diese generell optional sind. Solche Regeln werden uneingeschränkt ausgeführt. Die Funktionalität zur Überprüfung einer Bedingung ist im Block *APS Condition* gekapselt. Im Wesentlichen ist in der Bedingung ein boolescher Ausdruck definiert. Dieser besteht, aus Gründen der Effizienz und der Wiederverwertung von Berechnungen, aus mehreren Teilausdrücken (*APS Expression*), die in einer separaten Liste (*APS Expression List*) verwaltet werden. Dies hat den Vorteil, dass, falls mehrere Regeln evaluiert werden müssen,

gemeinsame Teilausdrücke jeweils nur einmal in einem Prüfvorgang berechnet werden müssen. Die Operanden der booleschen Ausdrücke sowohl in den Teilausdrücken als auch in den übergeordneten Bedingungen können entweder konstante numerische Ausdrücke sein, oder auf Werte des Datenpools (APS Data Pool), in der Regel Datensignale (*APS Data Signal*), verweisen. Damit lassen sich in den Regeln komplexe Bedingungen formulieren. Im Falle einer positiven Auswertung kann die Regel in einem letzten Schritt ausgeführt werden. Die Funktionalität hierzu ist in dem Block *APS Execution* ausgelagert. Wie bereits oben erwähnt wurde, besteht das Ziel eines Entscheiders bzw. seines Prüfers darin, Anweisungen zum Erstellen, Ändern oder Abbrechen von Aktivitäten und Aktionen zu generieren. Diese Handlungen sind in separaten, vom Block *APS Operation* abgeleiteten konkreten Blöcken (z.B. *APS Operation Activity Create* oder *APS Operation Action Destroy*), umgesetzt. Wie im Falle der Ausdrücke, können auch bei der Definition von Operationen Werte aus dem Datenpool verwendet werden. Sämtliche Operationen werden in einer separaten Liste (*APS Operation List*) verwaltet. Eine Ausführung lässt sich auf diese Weise mit mehreren Operationen verknüpfen. Somit könnte zum Beispiel eine Regel bei der Ausführung eine Gruppe von Aktivitäten desselben Typs zerstören, und eine alternative Aktivität in die Planung einfügen.

7.2.5.5 Ablaufplanung

Die Ablaufplanung bildet im autonomen Planungsprozess die Brücke zwischen der Entscheidungsfindung und der Planausführung. Sie orientiert sich an den in Abschnitt 2.3 erläuterten Prinzipien. Im Vordergrund dieses Abschnittes steht daher die Berechnung einer zeitlichen Abfolge von aus Aktivitäten abgeleiteten Aktionen, die zusammengefasst mit dem berechneten zukünftigen Verlauf der Ressourcen den eigentlichen Plan (schedule) ergeben. Die Ablaufplanung ist ein stark datenverarbeitender Prozess, der in einem eigens dafür vorgesehenen RODOS-Thread parallel zur Hauptsteuerung läuft. Die nachfolgende Abbildung 7-28 zeigt den dabei stattfindenden Datenfluss an den Prozessgrenzen in vereinfachter Form.

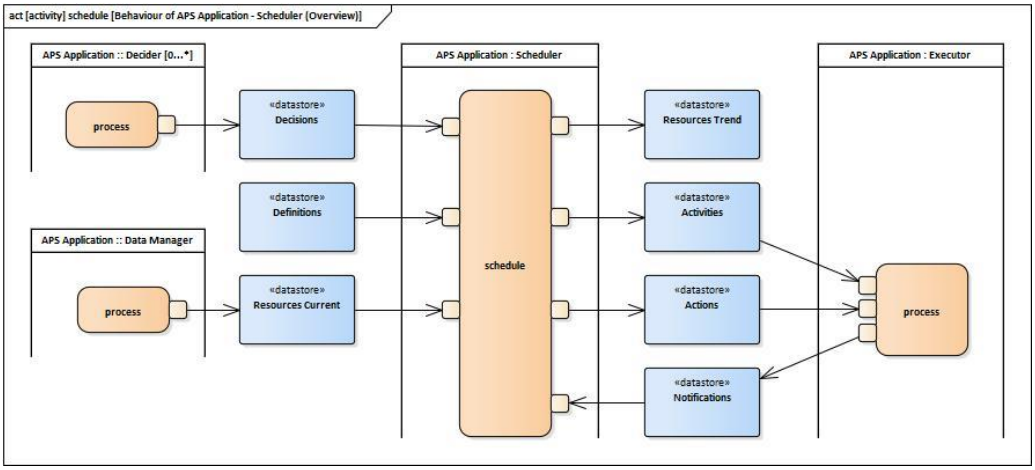


Abbildung 7-28: Interaktionen zur Ablaufplanung (Übersicht)

Die Hauptaufgabe des Ablaufplaners besteht darin, einen Plan in Form von Aktivitäten (Activities) und zeitlich geordneten Aktionen (Actions) unter Einhaltung von Rahmenbedingungen, zum Beispiel bezüglich der zur Verfügung stehenden Zeit und Ressourcen, zu entwerfen. Dabei werden auch Trends (Resources Trend) erstellt, die den zukünftigen zeitlichen Verlauf der Ressourcen entsprechen. Die Ausführung des Plans wird durch eine andere Instanz, dem Planausführer (Executor), realisiert. Dieser sendet rückmeldend dem Ablaufplaner Informationen (Notifications) über den Vorgang. Somit nimmt der Ablaufplaner auch eine Kontrollfunktion wahr, indem er Abweichungen bei der Ausführung des Plans feststellt und dementsprechend reagiert. Die Ablaufplanung ist ein dynamischer Prozess. Zwar wird das Verhalten des Ablaufplaners und damit die Art und Weise, wie Ablaufpläne erstellt werden, im Rahmen der Konfiguration des APS durch Definitionen (Definitions)

statisch vorgegeben, jedoch reagiert der Ablaufplaner auch dynamisch zur Laufzeit auf Eingangsimpulse in Form von umzusetzenden Entscheidungen (Decisions) und dem aktuellen Stand der Ressourcen (Resources Current).

Die nachfolgende Abbildung 7-29 zeigt beispielhaft eine schematische Darstellung eines Plans (ohne Ressourcen Trends).

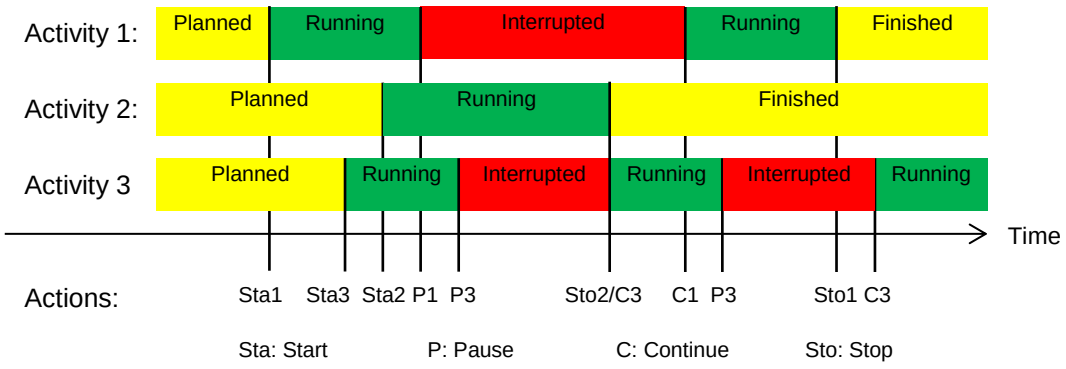


Abbildung 7-29: Schematische Darstellung eines Plans

Eine Aktivität kann, vereinfacht betrachtet, während der Ablaufplanung und der Planausführung (siehe Abschnitt 7.2.5.6) folgende Zustände einnehmen (siehe Abbildung 7-30):

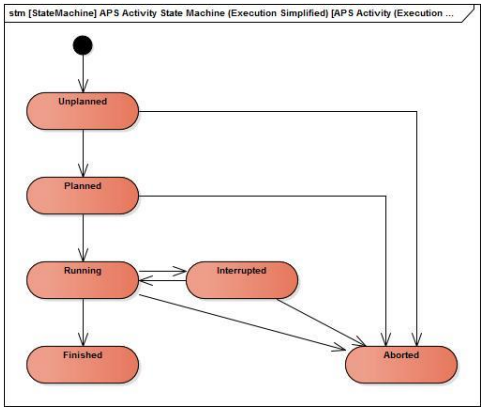


Abbildung 7-30: Zustandsmaschine für Aktivitäten (vereinfachte Darstellung)

Im Rahmen der Planausführung wird jeder Zustandsübergang einer Aktivität durch eine Aktion ausgelöst:

- Starten der Aktivität (Start)
- Pausieren der Aktivität (Pause)
- Fortführung der Aktivität (Resume)
- Stoppen der Aktivität (Stop)
- Abbruch der Aktivität (Abort)

Der geplante Verlauf einer Aktivität lässt sich auch in aktive und passive Intervalle unterteilen. Ein Intervall wird durch zwei Aktionen begrenzt:

- Start – Stop (aktiv)
- Start – Pause (aktiv)
- Fortführen – Pause (aktiv)
- Fortführen – Stop (aktiv)
- Fortführen – Abbruch (aktiv)

- Pause – Fortführen (passiv)

Neben der Zustandsmaschine besitzt eine Aktivität ihre durch die Entscheidungsfindung festgelegten Rahmenbedingungen zur Ablaufplanung:

- Frühester Startzeitpunkt der Aktivität (Release)
- Spätester Endzeitpunkt der Aktivität (Deadline)
- Minimale Dauer (nur aktiv) der Aktivität (DurationMin)
- Maximale Dauer (aktiv und passiv) der Aktivität (DurationMax)
- Unterbrechungen der Aktivität erlaubt (Interruptible) – Ja/Nein
- Punkte bei Abschluss der Aktivität (ScoreExecuted)
- Punkte pro 1 Sekunde aktive Laufzeit der Aktivität (ScoreActive)

Des Weiteren müssen auch Ressourcen in die Ablaufplanung eingebunden werden. Wenn entsprechende Definitionen vorliegen, können Ressourcen ebenfalls Rahmenbedingungen bei der Ablaufplanung darstellen. Die statischen Zusammenhänge werden mit Hilfe der nachfolgenden Abbildung 7-31 erklärt.

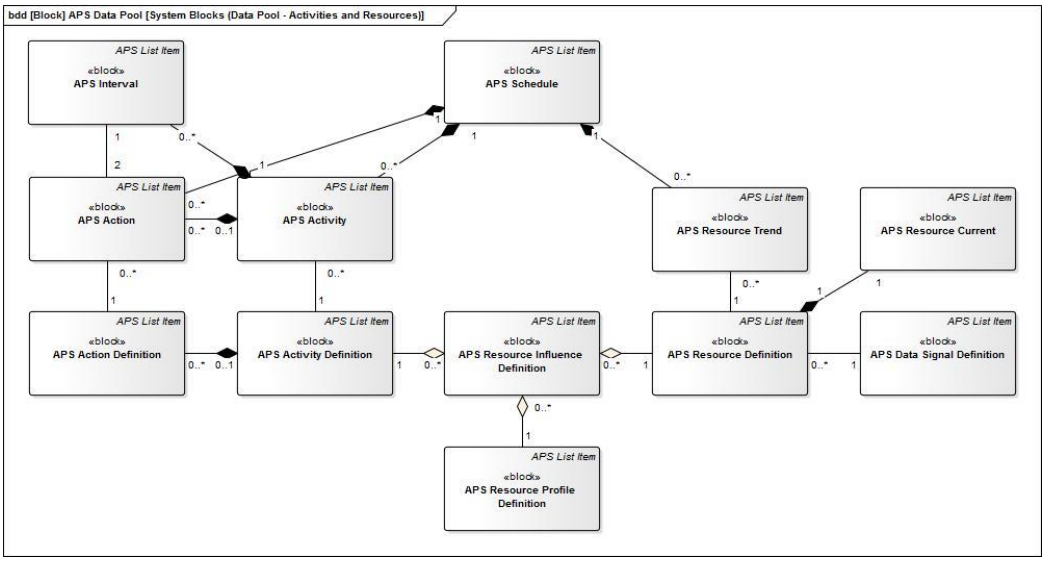


Abbildung 7-31: An der Ablaufplanung beteiligte Blöcke des Daten-Pools

Wie bereits erwähnt wurde, hat der Ablaufplaner zum Ziel, einen gültigen Plan (*APS Schedule*) zu erstellen. Dieser besteht aus Aktivitäten (*APS Activity*), Aktionen (*APS Action*), Intervallen (*APS Interval*) und Ressourcentrends (*APS Resource Trend*). Damit der Ablaufplaner gültige Pläne erstellen kann, benötigt er noch zusätzliches Wissen, das beschreibt, welche Aktivitäten und Ressourcen generell in der Planung existieren und wie sich diese gegenseitig beeinflussen. Diese Information wird im Rahmen der Konfiguration (siehe Abschnitt 7.2.5.1) eingelesen. Im Grunde handelt es sich um Definitionen (*APS * Definition*), die ein Szenario beschreiben. Dementsprechend existieren Definitionen für Aktivitäten (*APS Activity Definition*), Aktionen (*APS Action Definition*), Ressourcen (*APS Resource Definition*) und Beeinflussungen, gegeben durch Ressourcen-Auswirkungen (*APS Resource Influence Definition*) für die logische Verknüpfung der Aktivitäten und der Ressourcen zueinander sowie Ressourcen-Profilen (*APS Resource Profile Definition*) für die Kapselung der mathematischen Funktionen zur Berechnung der Ressourcentrends. Zusätzlich ist noch definitionsmäßig jede Ressource (*APS Resource Definition*) mit einem Datensignal (*APS Data Signal Definition*) verknüpft. Dadurch kann der Ablaufplaner die aktuellen Werte einer Ressource (*APS Resource Current*) aus den vom Datenmanager verwalteten Signalen ableiten. Jede Ressourcendefinition gibt zulässige Ober- und Untergrenzen für die entsprechende Ressource vor. Der Ablaufplaner muss bei der Planung stets sicherstellen, dass der Trend sich innerhalb dieser Beschränkung befindet, andernfalls ist der berechnete Plan ungültig.

Die Auswirkungen von Aktivitäten auf Ressourcen werden mit Hilfe von Profilen (*APS Resource Profile Definition*) berechnet. Mathematisch betrachtet, definiert ein Profil eine Funktion, die den zukünftigen Trend einer Ressource ermittelt. Ist eine Ressource mit mehreren Aktivitäten verknüpft, wird das Superpositionsprinzip angewendet, um den Gesamttrend für die entsprechende Ressource zu berechnen.

$$Profilfunktion\ p: (AD, RD, RO, t) \rightarrow \mathbb{Z}$$
$$AD: Definition\ einer\ Aktivität$$
$$RD: Definition\ einer\ Ressource$$
$$RO: Offsetwert\ einer\ Ressource$$
$$t: Zeit$$

Bei den Eingangsparametern können die Definitionen direkt über die entsprechenden Verknüpfungen der Aktivität und der Ressource abgeleitet werden. Als Offsetwert gilt der zum Zeitpunkt der Planerstellung festgehaltene Wert der Ressource (*APS Resource Current*). Die Zeit ist der laufende Parameter zur Berechnung des Trends.

Die nachfolgende Abbildung 7-32 zeigt eine Möglichkeit, wie ein Profil definiert sein könnte. Bei diesem Beispiel sind die verschiedenen Aktions- und Intervalltypen einer Aktivität von entscheidender Bedeutung.

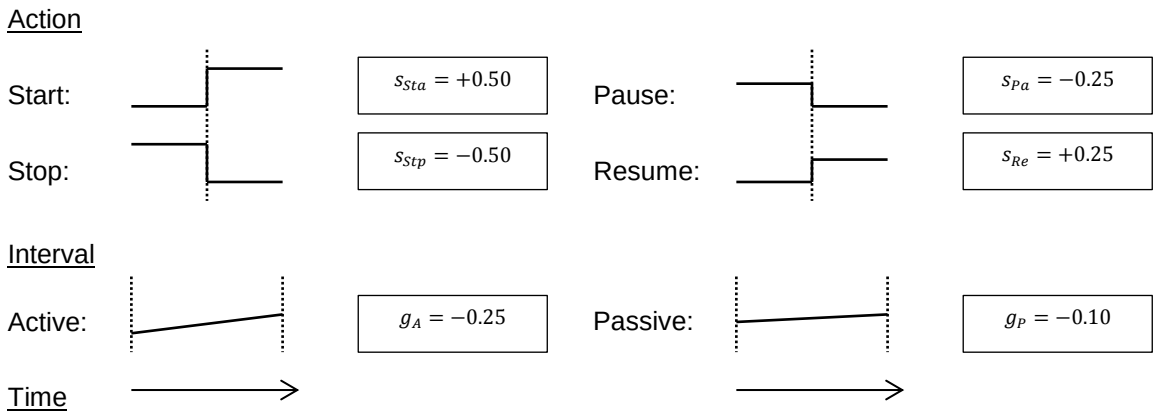


Abbildung 7-32: Definition eines Profils

Dieses Beispiel eines Profils definiert die Auswirkungen einer Aktivität auf die Ressource in Form von Sprüngen bei Zustandsänderungen und zusätzlich in Form von Integration während des Verweilens in einem Zustand (Intervall). Eine konkrete Auswirkung wird in der nachfolgenden Abbildung 7-33 gezeigt.

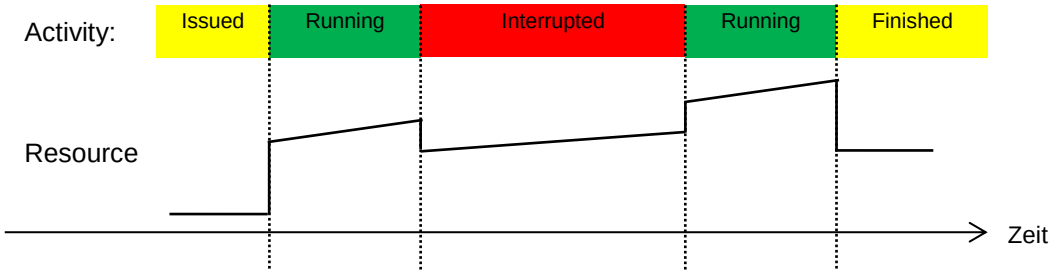


Abbildung 7-33: Anwendung eines Profils auf eine Ressource

An dieser Stelle sollte beachtet werden, dass dies nur ein einfaches Beispiel zur Veranschaulichung ist. In einem komplexeren Szenario könnten mehrere Verbrauchsarten von Ressourcen (*APS*

Ressource Influence Definition) definiert sein, die zur Folge hätten, dass sich mehrere Aktivitäten zur selben Zeit auf eine bestimmte Ressource auswirkten. Damit würde sich der Verlauf der Ressource mit mehreren Profilen überlagern.

Wie bereits zu Beginn des Abschnittes erwähnt wurde, besteht die Hauptaufgabe des Ablaufplaners darin, einen Plan bestehend aus Aktivitäten, Aktionen, Intervallen und Ressourcentrends zu generieren. Da sich die Ausgangssituation der Planung (z.B. Einbindung neuer Entscheidungen) laufend und auf unbestimmte Weise ständig ändern kann, muss der Planungsprozess kontinuierlich fortgesetzt werden. Eine weitere Schwierigkeit besteht darin, dass für den autonomen Betrieb zur jeder Zeit ein aktuell gültiger Plan zur Ausführung bereitstehen muss. Direkte Änderungen am aktuellen Plan dürfen jedoch nicht durchgeführt werden, da diese ihn unter Umständen korrumpieren können.

Des Weiteren müssen noch einige Sonderfälle in der Ablaufplanung berücksichtigt werden:

- Entscheider können die Planung von Aktionen, die in keinem Bezug zu einer Aktivität stehen, beauftragen. In der Regel steht dahinter die Absicht, Kommandos direkt in die Missionsplanung einzubinden.
- Es können Fehlerfälle auftreten. Diese deuten auf eine fehlerhafte Konfiguration des Ablaufplaners hin. Zum Beispiel können Ressourcen einen unzulässigen Wert (außerhalb der Min- und Max-Begrenzungen) einnehmen.
- Es können Ausnahmefälle eintreten. Diese deuten auf eine fehlerhafte Implementierung des Ablaufplaners hin. Zum Beispiel kann der Ablaufplanungsalgorithmus einen angeblich gültigen Plan ausgeben, der aber tatsächlich fehlerbehaftet ist.

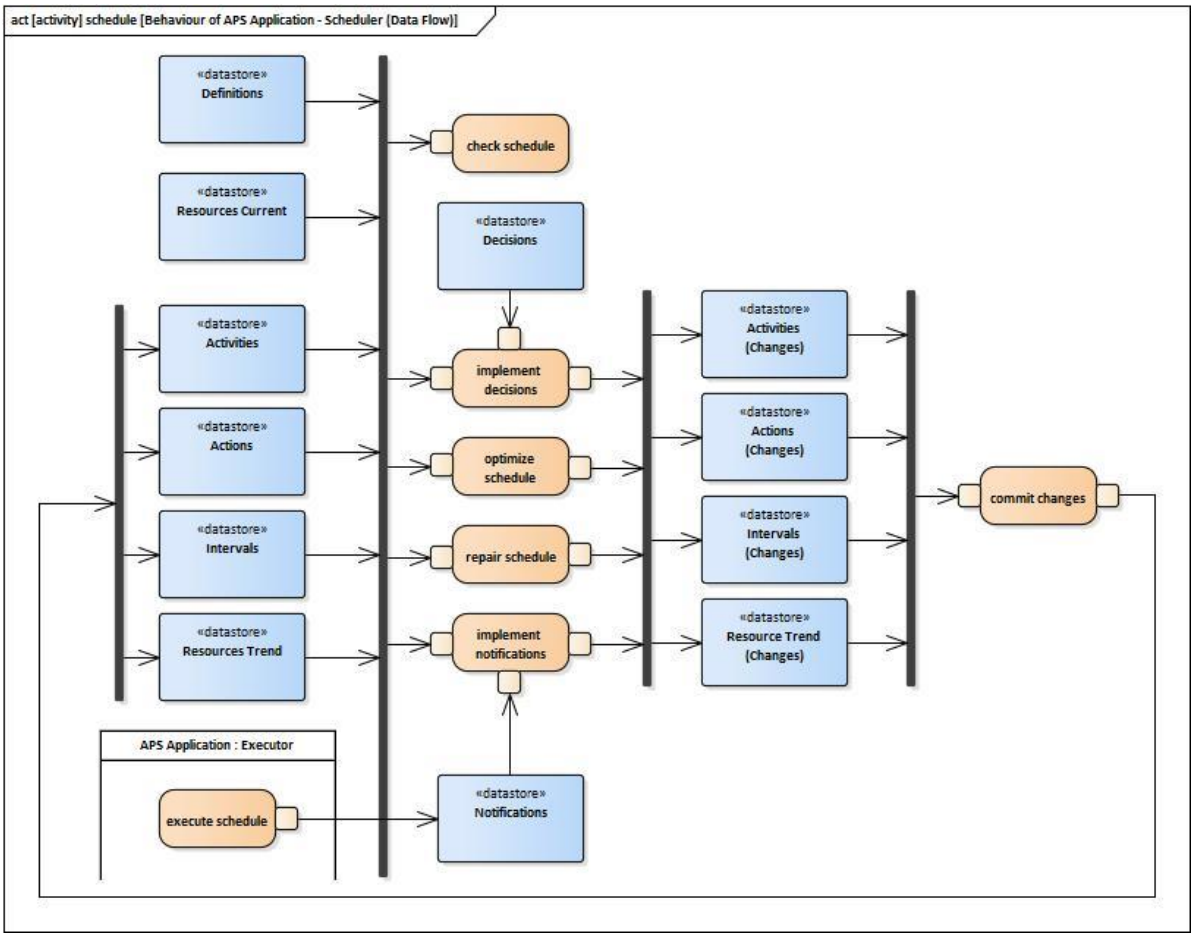


Abbildung 7-34: Interaktionen zur Ablaufplanung (Datenfluss)

In Fehler- und Ausnahmefällen ist vorgesehen, dass der Ablaufplaner die Planung einstellt und in einen Fehlerzustand übergeht. Dabei wird eine letzte (Fehler-)Aktion ausgegeben, die der Executor mit Anweisungen für die allgemeine Fehlerbehandlung (Safe-Mode) des Satelliten verknüpfen kann (siehe Abschnitt 7.2.6.2).

Die Ablaufplanung muss somit folgende Aufgaben erfüllen:

- Vom aktuellen Plan isolierte Durchführung und Evaluierung von Änderungen
- Übernahme von Planänderungen in den aktuellen Plan
- Einbindung der von Entscheidern getroffenen Entscheidungen in die Planung
- Laufende Optimierung des Plans
- Laufende Überprüfung des Plans hinsichtlich seiner Gültigkeit
- Einbindung von Rückmeldungen seitens der Planausführung
- Lösung von Konflikten (z.B. zu wenig Ressourcen, Fehlverhalten)

In zwei Abbildungen soll präsentiert werden, wie der Ablaufplaner die oben gestellten Aufgaben löst. Abbildung 7-34 zeigt hierbei den Datenfluss und Abbildung 7-35 den Kontrollfluss.

Die oben genannten Aufgaben werden im Ablaufplanungsprozess in jeweils eigene Funktionalitäten (*implement decisions, optimize schedule, repair schedule, implement notifications, check schedule, commit changes*) gekapselt. Diese wirken sich auf den laufend zu erstellenden Plan, bestehend aus Aktivitäten, Aktionen, Intervallen und Ressourcentrends, aus. Besonders auffallend ist dabei die Tatsache, dass zwei Bereiche existieren, in denen der Plan verwaltet wird. Der eine Bereich enthält den aktuell gültigen Plan, der momentan durch den *Executor* ausgeführt wird, der andere dient als eine Kopie, um mögliche Änderungen am Plan durch die verschiedenen Funktionalitäten des Ablaufplaners vorläufig zwischen zu speichern. Erst wenn sichergestellt werden kann, dass die Änderungen zulässig sind, werden diese durch eine spezielle Abgleichfunktion (*commit changes*) in den aktuell auszuführenden Plan übernommen. Andernfalls werden so lange Änderungen vorgenommen, bis der Konflikt gelöst wurde.

Eingangsseitig werden im Ablaufplanungsprozess dabei folgende Informationen verarbeitet:

- Definitionen (*Definition*) für den Ablaufplaner
- Entscheidungen (*Decisions*)
- Aktuelle Werte der Ressourcen (*Resources Current*)
- Rückmeldungen (*Notifications*) der Planausführung

Die Umsetzung von Entscheidungen (*implement decisions*) ist die elementarste Aufgabe des Ablaufplaners. Dieser Schritt ermöglicht es, Änderungswünsche von außen, d.h. durch die Entscheider und damit auch indirekt durch den Operator, in die Planung einzubeziehen. Die verschiedenen Arten von Entscheidungen, die es umzusetzen gilt, wurden bereits bei der Entscheidungsfindung vorgestellt (siehe Abschnitt 7.2.5.4) und werden nachfolgend kurz zusammengefasst:

- Hinzufügen, Ändern und Entfernen von Aktivitäten bzgl. der Planung bei Vorgabe der Rahmenbedingungen (z.B. Release und Deadline)
- Hinzufügen, Ändern und Entfernen von Aktionen (ohne direkten Bezug zu einer Aktivität) bzgl. der Planung bei Vorgabe der Rahmenbedingungen

In der Regel sollen neue Aktivitäten mit vorgegebenen Rahmenbedingungen in die Planung aufgenommen werden. Bei der Umsetzung von Entscheidungen muss eine Besonderheit betrachtet werden: Es kann der Fall eintreten, dass eine Entscheidung mit besonders hoher Priorität unmittelbar umgesetzt werden soll. Um unnötige Verzögerungen bis zur Ausführung der Entscheidung zu vermeiden, besteht die Möglichkeit, den aktuellen Plan - wenn nötig komplett - zu verwerfen, um die hochpriorisierte Entscheidung mit möglichst geringem Rechenaufwand in einen neuen Plan einzubinden. Die zuvor entfernten Aktivitäten können im Anschluss schrittweise in den neuen Plan

eingebunden werden. Hierbei kann es passieren, dass wieder einzufügende Aktivitäten im Konflikt zu der neuen hochpriorisierten Aktivität stehen. In diesem Fall erfolgt eine endgültige Verwerfung aus dem Plan. Die geopferte Aktivität kann aber im Anschluss durch die normal vorgesehene Ablaufplanung erneut in den Plan aufgenommen werden.

Der Ablaufplaner nimmt auch eine Kontrollfunktion wahr und kann im Rahmen einer Überprüfung (*check schedule*) feststellen, dass der aktuelle Plan inkonsistent geworden und damit in Zukunft nicht mehr ausführbar ist. Der Grund hierfür liegt meistens darin, dass die Erstellung eines Plans auf modellierten Grundlagen (z.B. Trends von Ressourcen anhand von Ressourcenprofilen) beruht, die jedoch von der Realität zu stark abweichen. Beispielsweise werden Ressourcen während der Planausführung viel stärker verbraucht, als im Plan vorgesehen. Der Ablaufplaner erkennt solche Situationen im Voraus und versucht diese Konflikte anschließend zu lösen (*repair schedule*). Die Planausführung muss im Falle eines zukünftigen erkannten Konfliktes nicht notwendigerweise unterbrochen werden. Der Plan wird vorläufig fortgeführt, aber zur gleichen Zeit auch korrigiert. Nur für den Fall, dass ein Konflikt zeitlich unmittelbar bevorsteht, muss die Ausführung des Plans unterbrochen werden. Des Weiteren kann der Fall eintreten, dass ein Plan nicht wie gewünscht durch den *Executor* ausgeführt werden kann. Der Ablaufplaner kann diesen Umstand feststellen, indem er die Rückmeldungen der Planausführung auswertet (*implement executions*). Zum Beispiel wird eine Aktivität vorzeitig beendet oder abgebrochen. Solche Planabweichungen müssen als äußere Störung und damit ebenfalls als Konflikt betrachtet und nach Möglichkeit durch den Ablaufplaner korrigiert werden.

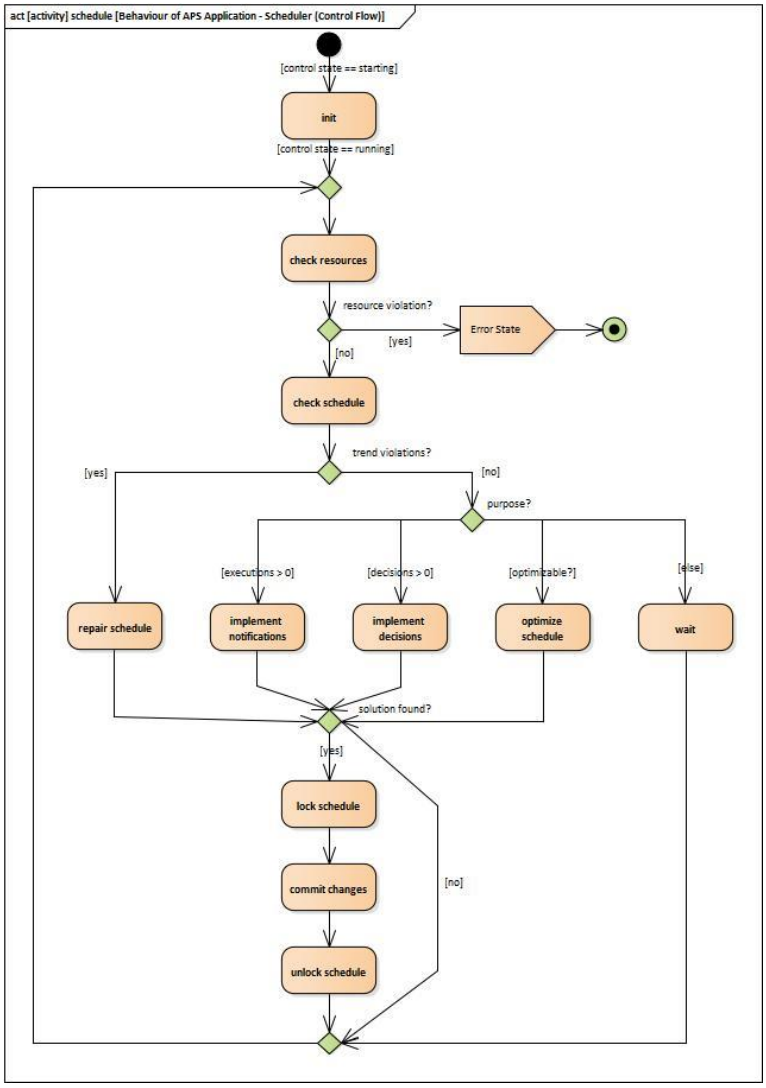


Abbildung 7-35: Interaktionen zur Ablaufplanung (Kontrollfluss)

Des Weiteren soll der Ablaufplaner nach Möglichkeit versuchen, den aktuellen Plan zu verbessern (*optimize schedule*). Zu den Rahmenbedingungen einer Aktivität gehören auch Angaben zu Punkten (ScoreExecuted, ScoreActive), die sie in die Bewertung des gesamten Plans einbringt. Das Optimierungskriterium des Ablaufplaners besteht darin, den Plan in dieser Hinsicht zu maximieren. Hierbei muss ein Kompromiss zwischen Qualität (Höhe der Punkte) und Laufzeit der Berechnungen gefunden werden. Die Optimierung ist nicht zeitkritisch, da zu diesem Zeitpunkt bereits ein gültiger Plan vorliegt, und läuft daher im Ablaufplanungsprozess mit niedriger Priorität. Um jedoch für die laufende Missionsplanung des Satelliten einen Nutzen ziehen zu können, sollte der Optimierungsprozess iterativ ablaufen. Das bedeutet, das Zwischenergebnisse, sofern diese einen verbesserten Plan darstellen, durchaus zwischenzeitlich in den aktuell gültigen Plan übernommen werden.

Die oben erwähnten Funktionalitäten (z.B. Umsetzung von Entscheidungen) können nicht alle gleichzeitig ablaufen, sondern müssen kontrolliert aufgerufen werden. Die Abbildung 7-35 zeigt den dabei stattfindenden Kontrollfluss.

Der Prozess des Ablaufplaners läuft, wie oben bereits erwähnt wurde, eigenständig in einem separaten RODOS-Thread und ist an einigen Stellen mit der Zustandsmaschine der Hauptsteuerung synchronisiert. Sobald der übergeordnete autonome Planungsprozess gestartet wird, beginnt der Ablaufplaner mit seiner eigenen Initialisierung und geht dann anschließend, wenn der operationale autonome Betrieb begonnen hat, in eine Kontrollschleife über. Diese läuft quasi endlos und terminiert nur (in Abbildung nicht dargestellt), wenn die Hauptsteuerung den autonomen Betrieb einstellt. Innerhalb der Schleife wird in einem Durchlauf nur eine der Funktionalitäten zur Lösung der oben genannten Teilaufgaben ausgewählt und abgearbeitet. Die Auswahl erfolgt anhand bestimmter Kriterien. Die Überprüfung der Ressourcen (*check resources*) und der aktuellen Pläne (*check schedule*) hat oberste Priorität und wird so oft wie möglich, jedoch mindestens vor jeder eigentlichen Berechnung zur Planänderung, durchgeführt. Hierbei werden Konflikte nur detektiert und festgehalten. Erst nachrangig kommen die Funktionalitäten, die tatsächlich Änderungen am Plan vornehmen, zum Einsatz. Mit abnehmender Priorisierung sind das die Behebung von zukünftigen Ressourcenüberschreitungen (*repair schedule*), die Verarbeitung der Rückmeldungen von der Planausführung (*implement notifications*) und die Umsetzung von Entscheidungen (*implement decisions*). Erst wenn solche Aufgaben nicht zur Abarbeitung anstehen, widmet sich der Ablaufplaner der Optimierung (*optimize schedule*), aber auch nur, wenn ein Anlass dafür besteht. Zum Beispiel wäre das der Fall, wenn durch eine vorherige Entscheidungsumsetzung Ressourcen plötzlich wieder frei würden, da eine Aktivität aus der Planung entfernt wurde. Sollten generell keine Änderungen am Plan nötig sein, auch nicht durch Optimierung, so sollte aus Gründen der Effizienz der Ablaufplaner seinen zugehörigen RODOS-Thread für eine bestimmte Dauer stilllegen (*wait*), und dadurch inaktiv werden.

Die bisher genannten Funktionalitäten der Ablaufplanung, zum Beispiel Optimierung, sind hierarchisch betrachtet auf der obersten Ebene anzusiedeln. Sie dienen zwar unterschiedlichen Zwecken, greifen jedoch auf gemeinsame Grundfunktionen zurück. Die nachfolgende Abbildung 7-36 zeigt schematisch die Zusammenhänge auf.

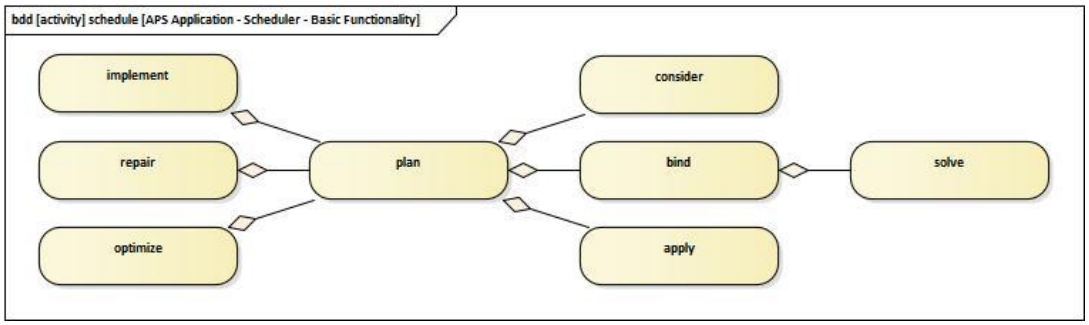


Abbildung 7-36: Funktionalitäten der Ablaufplanung und deren Zusammensetzung

Zu den Kernaufgaben der Ablaufplanung gehören die bereits bekannten Funktionalitäten:

- umsetzen (implement)
- reparieren (repair)
- optimieren (optimize)

Während ihrer Ausführung greifen sie im Rahmen der Planung (*plan*) auf grundlegendere Funktionalitäten zurück:

- abwägen (*consider*)
- binden (*bind*)
- anwenden (*apply*)

Die Planung koordiniert die dazu notwendigen Abläufe. Sie speichert Ausführungszustände und Zwischenergebnisse, um nach deren iterativen Übernahme in den aktuell gültigen Plan (*commit*) die Arbeit an alter Stelle wiederaufzunehmen.

Die Möglichkeiten, wie Änderungen an einem Plan vorgenommen werden können, sind begrenzt. Bisher wurden die folgenden Maßnahmen implementiert. Allen ist gemein, dass sie intuitiv verständlich sind:

- Hinzufügen ungeplanter Aktivitäten
- Entfernen geplanter, aber noch nicht ausgeführter Aktivitäten
- Abbruch geplanter und derzeit sich in der Ausführung befindender Aktivitäten
- Hinzufügen passiver Intervalle in einer Aktivität
- Entfernen passiver Intervalle aus einer Aktivität
- Zeitliche Verschiebung/Neuberechnung aller zu einer Aktivität gehörender Aktionen

Das Berechnen von Änderungen am Plan beginnt immer zuerst mit dem Abwägen von Planänderungen. Dafür werden mögliche Maßnahmen identifiziert. So müssen zum Beispiel im Falle einer Konfliktlösung die Maßnahmen den Konflikt beheben, bzw. schwächen. Im Falle einer Optimierung müssen die Maßnahmen den Punktestand des Plans erhöhen. Hierbei müssen die Auswirkungen der Maßnahmen auf die Ressourcen berücksichtigt werden. Ebenso gilt es die Randbedingungen (z.B. Release und Deadlines einer Aktivität) einzuhalten. Die Güte einer Maßnahme ist das Verhältnis zwischen dem Beitrag an Punkten und den Auswirkungen auf die Ressourcen. Anhand der Güte können die einzelnen Maßnahmen miteinander verglichen werden. Je nach Rechenaufwand der Maßnahme kann es sein, dass nur eine Schätzung für die Abwägung vorgenommen werden kann. Bei der Planung können je nach Konfiguration auch mehrere Maßnahmen kombiniert betrachtet, untersucht und angewendet werden.

Unter dem Binden wird das detaillierte Berechnen von Maßnahmen unter Einbeziehung der zuvor schon erwähnten Randbedingungen verstanden. Auch hierbei gilt es, jede Auswirkung einer Maßnahme auf eine Ressource, falls sie zur Anwendung kommt, zu berücksichtigen. Das Berechnen einer Bindung ist wesentlich aufwendiger als das Schätzen einer Abwägung. Jedoch liefert die Bindung ein exaktes Ergebnis. Beim Binden können auch mehrere Maßnahmen in Kombination berechnet werden.

In einem letzten Schritt kann, wenn eine gültige Bindung vorliegt, diese auf den Plan angewendet werden. Die darin enthaltenen Vorgaben werden dann exakt umgesetzt. Erst mit diesem Schritt hat die Maßnahme eine tatsächliche Auswirkung auf den Plan.

Das Binden wird dadurch erschwert, dass sich die Aktivitäten möglicherweise auf mehrere Ressourcen auswirken können. Es kann passieren, dass angewendete Maßnahmen zwar ihren Zweck erfüllen, dabei aber neue Konflikte verursachen. Als Konsequenz hieraus müssen weitere Maßnahmen berechnet werden, um die negativen Folgen zu beheben. Hilfreich ist an dieser Stelle,

dass das Problem der Bindung mathematisch als ein lineares Programm formuliert und separat durch einen passenden Algorithmus gelöst (*solve*) werden kann.

Ein lineares Programm [13, pp. 857 - 910] [53] stellt ein zu lösendes Problem (z.B. die Suche nach einer passenden Bindung) formal in folgender Form dar:

maximiere die Zielfunktion $c^T * x$

unter den Nebenbedingungen $A * x \leq b$

$$x \geq 0$$

Diese Darstellung kann als ein System aus Ungleichungen aufgefasst werden. x ist der zu bestimmende *m*-dimensionale Lösungsvektor, A eine *m x n* - dimensionale Matrix, b ein *m*-dimensionaler Spaltvektor und c^T ein n-dimensionaler Zeilenvektor. *m* bezeichnet hierbei die Anzahl der Nebenbedingungen und *n* die Anzahl der Lösungselemente.

Mit Hilfe des Simplex-Algorithmus [13, p. 885] lässt sich das System durch sogenannte Pivotisierungs-Operationen in eine andere äquivalente Form umformen. In der Regel erhöht sich dadurch der Wert der Zielfunktion. Diese Umformungen werden solange wiederholt, bis keine Verbesserung des Zielwertes mehr möglich ist und das Verfahren dadurch terminiert. Wenn eine Lösung gefunden wurde, dann ist diese, theoretisch betrachtet, exakt und optimal. Jedoch können bei der Berechnung numerische Fehler auftreten, die zu ungültigen Lösungen führen. Die Ursache hierfür liegt bekannterweise in der sukzessiven Aneinanderreihung der Pivot-Operationen und in der Fließkomma-Arithmetik (Rundungsfehler und Ungenauigkeiten). Um diese Fehler möglichst zu vermeiden, wurde der Simplex-Algorithmus, angelehnt an [53, p. 79], angepasst. Das sogenannte Simplex-Tableau, das die internen Zustandsdaten des linearen Programms enthält, kann bei Bedarf direkt aus den Original-Daten (A, b und c^T) rekonstruiert werden. Diese Vorgehensweise vermeidet die numerischen Fehler, die durch die Pivot-Sequenzen verursacht werden, größtenteils.

Um den Simplex-Algorithmus anwenden zu können, muss das Planungsproblem in ein lineares Programm (bestehend aus einem System aus Ungleichungen und der Zielfunktion) überführt werden. Der Lösungsvektor x stellt dabei die exakten Zeitpunkte dar, zu denen die Aktionen der Aktivitäten im Plan ausgeführt werden sollen. Zur Bildung des linearen Programms sind folgende Schritte notwendig:

1. Für jede zu planende Aktivität a muss der früheste Startzeitpunkt $Ta_{Release}$ berücksichtigt werden.

$$-1 * xa_{Start} \leq -1 * Ta_{Release}$$

2. Für jede zu planende Aktivität a muss der späteste Endzeitpunkt $Ta_{Deadline}$ berücksichtigt werden.

$$+1 * xa_{Stop} \leq +1 * Ta_{Deadline}$$

3. Für jede zu planende Aktivität a muss die Mindestdauer Ta_{Min} berücksichtigt werden. Diese Beschränkung bezieht sich nur auf die zur Aktivität a gehörenden aktiven Intervalle I_{aA}.

$$\sum_{i}^{I_{aA}} (+1 * xi_{Begin} - 1 * xi_{End}) \leq -1 * Ta_{Min}$$

   Falls als Maßnahme ein Abbruch der Aktivität a vorgesehen ist, darf diese Bedingung nicht in das System aufgenommen werden.

4. Für jede zu planende Aktivität a muss die zulässige Maximaldauer Ta_{Max} berücksichtigt werden. Alle zur Aktivität a gehörenden Intervalle I_a (aktive und passive) sind zu betrachten.

$$\sum_{i}^{I_a}(-1 * xi_{Begin} + 1 * xi_{End}) \leq +1 * Ta_{Max}$$

5. Für jede zu planende Aktivität a muss die zeitliche logische Ordnung aller ihrer Aktionen berücksichtigt werden. Hierzu muss für alle zur Aktivität a gehörenden Intervalle I_a jeweils eine Bedingung aufgestellt werden:

$$\forall i \in I_a : +1 * xi_{Begin} - 1 * xi_{End} \leq 0$$

6. Jede Aktion j, die bereits zu einem Zeitpunkt Tj ausgeführt wurde (d.h. die zugehörige Aktivität befindet sich derzeit in der Ausführung), muss zeitlich fixiert werden. Für jede Fixierung müssen zwei Bedingungen in das System aufgenommen werden. Der Simplex-Algorithmus wird dadurch gezwungen, den Zeitpunkt der Aktion j auf Tj zu setzen.

$$xj \leq +1 * Tj$$

$$-xj \leq -1 * Tj$$

7. Wenn die Optimierung aktiviert ist, muss sich der Punktestand des zu berechnenden Plans gegenüber der letzten gültigen Lösung verbessern. Alle Aktivitäten A können mit ihren aktiven Intervallen zur Gesamtbewertung des Plans beitragen, indem sie Punkte pro Zeit s_T und pro Ausführung s_S produzieren. Zusätzlich müssen alle Maßnahmen M, deren Komponenten nicht im Lösungsvektor abgebildet sind (z.B. Löschen von Aktivitäten), zusätzlich durch einen Offset s_M in die Gleichung aufgenommen werden.

$$\sum_{a}^{A} \sum_{i}^{I_{aA}}(+s_{Ta} * xi_{Begin} - s_{Ta} * xi_{End}) \leq -\left[\sum_{a}^{A}(+1 * s_{Sa}) + \sum_{m}^{M}(+1 * s_{Mm})\right]$$

   Falls eine Maßnahme vorsieht, dass eine Aktivität a abgebrochen werden sollte, so dürfen deren Beiträge zum Punktestand nicht mitgerechnet werden. In diesem Fall ist es ausreichend, die Werte für s_{Ta} und s_{Sa} auf 0 zu setzen.

8. Für jede Ressource $r \in R$ muss sichergestellt werden, dass der Verlauf die zulässigen Minimal- und Maximalwerte (v_{rMin} und v_{rMax}) nicht über- bzw. unterschreitet. Erschwerend kommt hinzu, dass auf Grund des Ressourcen-Profils mit jeder Aktion $j \in J$ der Verlauf einen Sprung T_j in Abhängigkeit von j vollziehen kann (siehe Abbildung 7-32 und Abbildung 7-33). Falls sich eine Aktivität $a \in A$ auf eine Ressource $r \in R$ auswirkt, muss daher für jede zu a gehörende Aktion $j \in J_a$, $J_a \subseteq J$ der eingehende und der ausgehende Ressourcen-Wert separat mit einer Bedingung gegen den Minimal- und den Maximalwert unter Zuhilfenahme der Intervalle $i \in I_a$ von a abgesichert werden. Es muss zwischen aktiven Intervallen I_A und passiven Intervallen I_P unterschieden werden, da sie sich unterschiedlich auf die Ressourcen (Ga_{ra} für aktive Phasen und Gp_{ra}für passive Phasen) auswirken. Beim Aufstellen der Bedingungen ist zu beachten, dass für eine beliebige Aktion j_K nur die zurückliegenden Aktionen einschließlich J_K in der Bedingung zu berücksichtigen sind (siehe Abbildung 7-37).

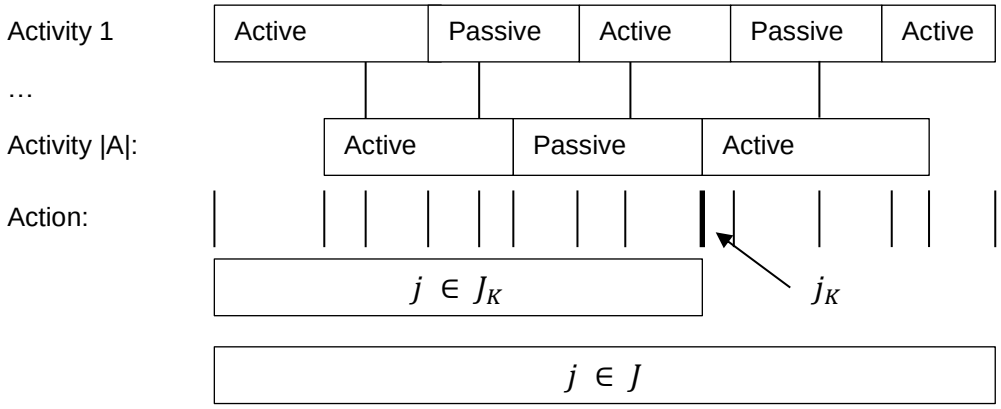


Abbildung 7-37: Zuordnung der Aktion j zu J_K

Zusammengefasst müssen für jede Ressource $r \in R$, die von Aktivitäten aus $A_r \subseteq A$ beeinflusst wird, jeweils vier Bedingungen pro Aktion j_a aus $j \in J_a$ aufgestellt werden, wobei $a \in A_r$ gilt.

1. $f_r \leq v_{rMax} - v_r - \Delta in_r$
2. $f_r \leq v_{rMax} - v_r - \Delta out_r$
3. $-1 * f_r \leq -1 * (v_{rMin} - v_r - \Delta in_r)$
4. $-1 * f_r \leq -1 * (v_{rMin} - v_r - \Delta out_r)$

f_r berechnet den linearen Anteil am Verlauf der Ressource r bis zur Aktion j_K. v_r stellt den aktuellen Wert der Ressource r dar.

$$f_r = \sum_{a}^{A} \left[\sum_{i}^{I_{aAc}} (+Ga_{ra} * xi_{Begin} - Ga_{ra} * xi_{End}) + \sum_{i}^{I_{aPc}} (+Gp_{ra} * xi_{Begin} - Gp_{ra} * xi_{End}) + \sum_{i}^{I_{aAo}} (+Ga_{ra} * xi_{Begin} - Ga_{ra} * xj_k) + \sum_{i}^{I_{aPo}} (+Gp_{ra} * xi_{Begin} - Gp_{ra} * xj_k) \right]$$

I_{aAc} bezeichnet dabei die Menge der aktiven Intervalle von der Aktivität a, mit der Eigenschaft, dass diese bis zum Zeitpunkt von j_k abgeschlossen (c für closed) sind. I_{aPc} bezieht sich entsprechend auf passive Intervalle. I_{aAo}, I_{aPo} gilt für offene (o für open) Intervalle:

$$I_{aAc} = \{i \in I_{aA} \mid xi_{Begin} < xj_k \wedge xi_{End} < xj_k\}$$

$$I_{aPc} = \{i \in I_{aP} \mid xi_{Begin} < xj_k \wedge xi_{End} < xj_k\}$$

$$I_{aAo} = \{i \in I_{aA} \mid xi_{Begin} < xj_k \wedge xi_{End} > xj_k\}$$

$$I_{aPo} = \{i \in I_{aP} \mid xi_{Begin} < xj_k \wedge xi_{End} > xj_k\}$$

Δin_r, Δout_r sind Größen, die den sprungbehafteten Anteil am Verlauf der Ressource r bis zur Aktion j_K summieren.

$$\Delta in_r = \sum_{j}^{J_k \setminus j_k} T_j$$

$$\Delta out_r = \sum_{j}^{J_k} T_j$$

Für den (aufwendigsten) Fall, dass sich jede im Plan vorkommende Aktivität $a \in A$ auf jede Ressource $r \in R$ auswirkt, müssen insgesamt $|R| * |J| * 4$ Bedingungen aufgestellt werden.

9. Damit das lineare Programm komplett aufgestellt ist, müssen die Koeffizienten der Zielfunktion $c^T * x$ gesetzt werden. Im Schritt 7 wurde bereits erläutert, auf welche Weise das System Punkte generiert. Als Zielfunktion gilt daher:

$$\sum_{a}^{A} \sum_{i}^{I_{aA}} (+s_{Ta} * xi_{Begin} - s_{Ta} * xi_{End})$$

Durch Anwendung der Schritte 1 bis 9 lässt sich das Ablaufplanungsproblem in ein System aus linearen Ungleichungen und einer Maximierungsfunktion umwandeln. In der Regel hat das resultierende System einige Bedingungen, die durch andere, schärfere Bedingungen bereits abgedeckt sind. Dies ist der Fall, wenn die Matrix A der Nebenbedingungen $A * x \leq b$ lineare Abhängigkeiten zwischen den Zeilen aufweist. Hier können die unschärferen Nebenbedingungen ausnahmslos entfernt werden. Dadurch sinkt die Komplexität des aufgestellten Systems, und folglich auch der benötigte Aufwand zur Berechnung der Lösung.

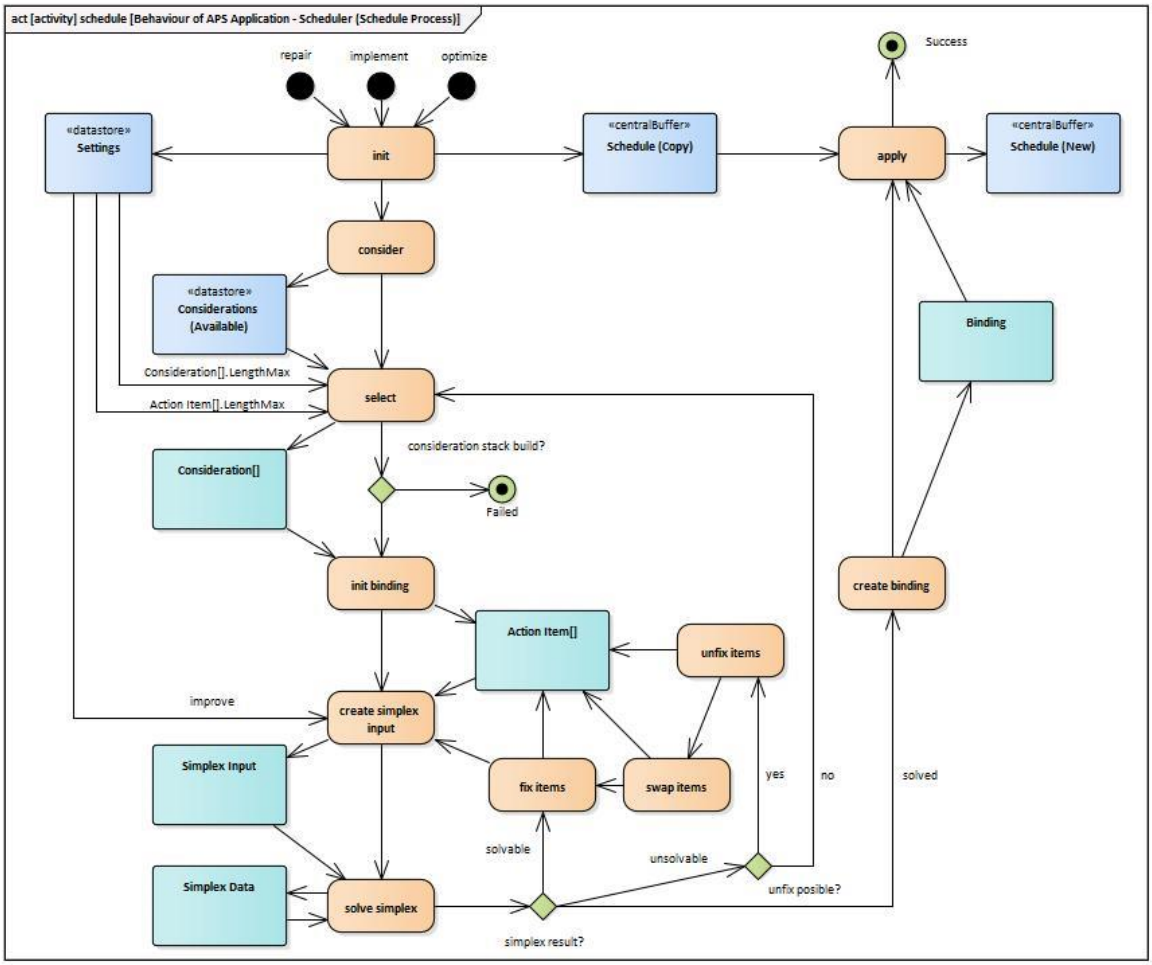


Abbildung 7-38: Auflösen einer Abwägung bei der Ablaufplanung

Bisher wurden die in Abbildung 7-36 dargestellten Funktionalitäten, angefangen bei den Grundfunktionalitäten, z.B. die Optimierung, über die Abwägung und Bindung bis hin zur Auflösung (*solve*) isoliert betrachtet. Abbildung 7-38 zeigt nun den Vorgang der Ablaufplanung in seinem Zusammenwirken aller Teilfunktionalitäten.

Die Ablaufplanung wird im Rahmen einer der Hauptfunktionalitäten (reparieren, implementieren oder optimieren) eingeleitet. Zu Beginn (*init*) wird der aktuell gültige Plan kopiert. Die Kopie bildet die Grundlage für alle nachfolgenden Schritte und wird im weiteren Verlauf durch diese verändert. Ferner wird eine Strategie, die die weitere Vorgehensweise der Ablaufplanung festlegt, in den Einstellungen (*Settings*) hinterlegt. Sie hat Auswirkungen auf das Verhalten beim Abwägen (*consider*), beim Binden (*bind*) und beim Lösen (*solve*). Die Strategien verfolgen folgende Absichten:

- Reparieren: Es soll so schnell wie möglich ein gültiger Plan wiederhergestellt werden. Der neue (gültige) Plan darf bzgl. der Punkte schlechter sein als der alte (ungültige) Plan.
- Implementieren: Es soll so schnell wie möglich eine anliegende Entscheidung in den neuen Plan umgesetzt werden. Der neue Plan muss bzgl. der Punkte besser sein als der alte Plan.
- Optimieren: Es soll der beste Plan ermittelt werden. Der Plan darf iterativ verbessert und übernommen werden.

Zuerst werden alle Maßnahmen (z.B. Einfügen einer neuen Aktivität), die für die Lösungsfindung und für die zugrundeliegende Strategie nützlich sind, identifiziert. Generell dürfen bereits im Plan vorhandene Aktionen angepasst bzw. verschoben werden. In Abhängigkeit von der Strategie werden weitere Maßnahmen in Betracht gezogen.

- Reparieren: Nur Löschen oder Abbrechen bereits vorhandener Aktivitäten.
- Implementieren: Nur die der umzusetzenden Entscheidung zugrundeliegende Maßnahme. In der Regel das Einfügen neuer Aktivitäten. Laut Strategie dürfen nur Maßnahmen in Betracht gezogen werden, die den Wert (Punkte) des neuen Plans erhöhen.
- Optimieren: Grundsätzlich alle möglichen Maßnahmen.

Als nächstes ist eine Kombination aus Maßnahmen auszuwählen (*select*) und in einem Maßnahmenstapel (*Consideration[]*) festzuhalten. Kombinationen sind dann nötig, wenn die gewinnbringende Maßnahme auf nicht vorhandene Ressourcen angewiesen ist, die durch andere Maßnahmen zuerst beschafft werden müssen. Über die Einstellungen kann die maximale Tiefe des Stapels festgelegt werden. Dieser Parameter ist sehr kritisch, da auf Grund des kombinativen Charakters die Laufzeit mit steigender Stapelgröße exponentiell ansteigt. In der Regel müssen mehrere Kombinationen von Maßnahmen ausprobiert werden, ehe eine Lösung gefunden wird. Wenn der gesamte Suchraum durchlaufen wurde, also wenn keine weiteren Kombinationen an lösbaren Maßnahmen identifiziert werden können, ist das konkrete Ablaufplanungsproblem (in der Regel die Optimierung) mit den dem Ablaufplaner zur Verfügung stehenden Möglichkeiten (z.B. Maßnahmen und Stapeltiefe) nicht lösbar. In Folge dessen wird der gesamte Vorgang abgebrochen (*failed*).

Konnte hingegen eine möglich passende Kombination von Maßnahmen ausgewählt werden, kann mit der eigentlichen Bindung begonnen werden. Diese hat zu Beginn die Aufgabe, ein Feld von Aktionsitems (*Action Item[]*) zu initialisieren. In diesem wird die zeitliche Ordnung aller im Plan vorkommenden Aktionen definiert. Am Plan durch Maßnahmen vorgegebene Änderungen sind ebenfalls zu berücksichtigen (z.B. zwei neue Aktionsitems für eine neu einzufügende Aktivität). Der Bindungsablauf legt damit schon vor dem eigentlichen Aufruf des Simplex-Algorithmus (*solve simplex*) die logische Reihenfolge aller im Plan vorkommender Aktionen fest. Der Simplex-Algorithmus muss diese Vorgaben der Ordnung (in Form von zusätzlichen Bedingungen) einhalten, wenn er eine gültige Lösung ermitteln soll. Falls für eine konkrete Ordnung keine gültige Lösung ermittelt werden kann, ist es dem Ablaufplaner im Rahmen der Bindung erlaubt, die Reihenfolge durch Tauschoperationen (*swap*) zu verändern und somit Alternativen auszuprobieren. Somit können theoretisch alle möglichen Varianten (bzw. Permutationen der *Action Item[]*) untersucht werden. Diese Vorgehensweise der vorzeitigen Definition der zeitlichen Ordnung ist notwendig, da sich sonst die Bedingungen für die

Ressourcenbeschränkungen (siehe Schritt 8, Aufstellung Simplex-Problem) nicht aufstellen lassen. Im Grunde wird dadurch das eigentliche nicht-lineare Ablaufplanungsproblem (wegen den möglichen Sprüngen im Verlauf einer Ressource, siehe Ressourcenprofil) in mehrere lineare Varianten aufgeteilt und in lineare Programme (*Simplex Input*) für den Simplex-Algorithmus (*solve simplex*) umgeformt. Aufgrund des kombinatorischen Charakters des Problems steigt der Suchraum exponentiell mit der Anzahl der im Plan zu berechnenden Aktionen. Dieser Suchraum lässt sich, noch vor der Anwendung des Simplex-Algorithmus, durch Analyse der zu den Aktivitäten gegebenen Rahmenbedingungen (z.B. Release und Deadline) stark einschränken. Im Grunde werden unlösbare Varianten aussortiert. Ein weiterer Mechanismus zur Steigerung der Effizienz der Ablaufplanung besteht darin, ganze Gruppen von Varianten möglichst frühzeitig als unlösbar zu identifizieren, ohne dabei alle Rahmenbedingungen für den Simplex-Algorithmus aufzustellen. Zur Erklärung ist es hilfreich, den Suchraum als eine Baum-Struktur (siehe Abbildung 7-40) zu betrachten, der bei der Suche nach einer Lösung traversiert wird. Dabei wird die Ordnung der Aktionen (*Action Item[]*) mit zunehmender Tiefe schrittweise fixiert (siehe Abbildung 7-39).

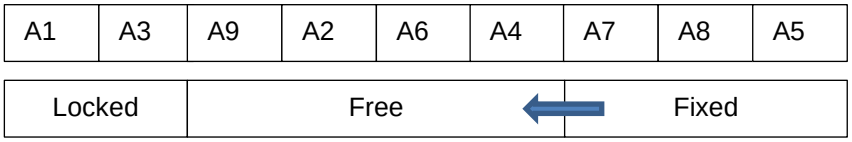


Abbildung 7-39: Fixieren der Ordnung von Aktionen (Items)

Jede zu planende Aktion kann einer der folgenden drei Gruppen zugeordnet werden:

- Locked: Zu dieser Gruppe gehören Aktionen, die schon ausgeführt wurden. Es werden daher Bedingungen entsprechend Schritt 6 zur Aufstellung des linearen Programms gesetzt.
- Free: Zu dieser Gruppe gehören Aktionen, die nicht fixiert sind. Für alle Aktionen, die dieser Gruppe angehören, werden die Bedingungen entsprechend Schritt 8 zur Aufstellung des linearen Programms nicht gesetzt.
- Fixed: Zu dieser Gruppe gehören Aktionen, die fixiert sind. Für alle Aktionen, die dieser Gruppe angehören, werden die Bedingungen entsprechend Schritt 8 zur Aufstellung des linearen Programms gesetzt.

Der Schritt 8 zur Aufstellung des linearen Programms ist der aufwendigste. Er beschreibt, welche Bedingungen für den Simplex-Algorithmus gesetzt werden müssen, um die Gültigkeit des Ressourcenverlaufs zu gewährleisten. Zu Beginn der Traversierung ist jede Aktion entweder der Gruppe „Locked“ oder der Gruppe „Fixed“ zugeordnet. Dadurch verringert sich der Aufwand zur Berechnung der „gelockerten“ linearen Programme. Der Simplex-Algorithmus kann neben der Optimierungs-Berechnung auch zur ausschließlichen Lösbarkeits-Berechnung (ohne Optimierung) eines linearen Programms herangezogen werden. Dieser Schritt erfordert weniger Rechenschritte als die komplette Berechnung des Optimums. Falls der Simplex-Algorithmus die Lösbarkeit eines linearen Programms mit nicht-leerer Free-Gruppe feststellt, kann die Bindung das Problem verschärfen und eine Fixierung (*fix*) vornehmen. Dabei wechselt die Aktion aus der Gruppe „Free“, die der Gruppe „Fixed“ zeitlich gesehen durch die Ordnung an nächsten liegt, in die Gruppe „Fixed“. Dieser Schritt der Fixierung wiederholt sich solange, bis alle Aktionen aus der Gruppe „Free“ entfernt wurden. Im Struktur-Baum ist die Traversierung dann an einem (grünen) Blatt angelangt und hat eine Lösung gefunden. Diese muss vom Simplex-Algorithmus hinsichtlich des Optimums (und nicht nur der Lösbarkeit) berechnet werden. Stellt hingegen der Simplex-Algorithmus die Unlösbarkeit des linearen Programms während der Traversierung fest (roter Knoten/Blatt), kann die Überprüfung des aktuellen Baumzweiges beendet werden, da die nachfolgenden Verzweigungen ebenfalls unlösbar sind. Mit zunehmender Tiefen-Traversierung verschärfen sich bekanntlich nur die Bedingungen des linearen Programms. Die Traversierung kann dann wieder in Richtung Wurzel des Struktur-Baums zur letzten lösbaren Verzweigung aufsteigen, dort die letzte Fixierung wieder aufheben (*unfix*, siehe Abbildung 7-38) und die Ordnung der Aktionen (der freien Menge) mit einer Vertauschung (*swap*) verändern.

Dabei springt die Traversierung am Struktur-Baum horizontal und fährt anschließend wieder in Richtung Blätter (entspricht Fixierung). Wenn bei einem Knoten keine Vertauschungen mehr möglich sind, muss die Traversierung ebenfalls einen Knoten aufsteigen.

Die Vorgehensweise bei der Traversierung hat den Vorteil, dass ganze Zweige des Such-Baums (siehe Abbildung 7-40) bei Feststellung einer unlösbaren Verzweigung von der Suche ausgeschlossen werden. Andernfalls müsste enorme Rechenleistung in die Durchforstung des Suchraums nach einer Lösung investiert werden.

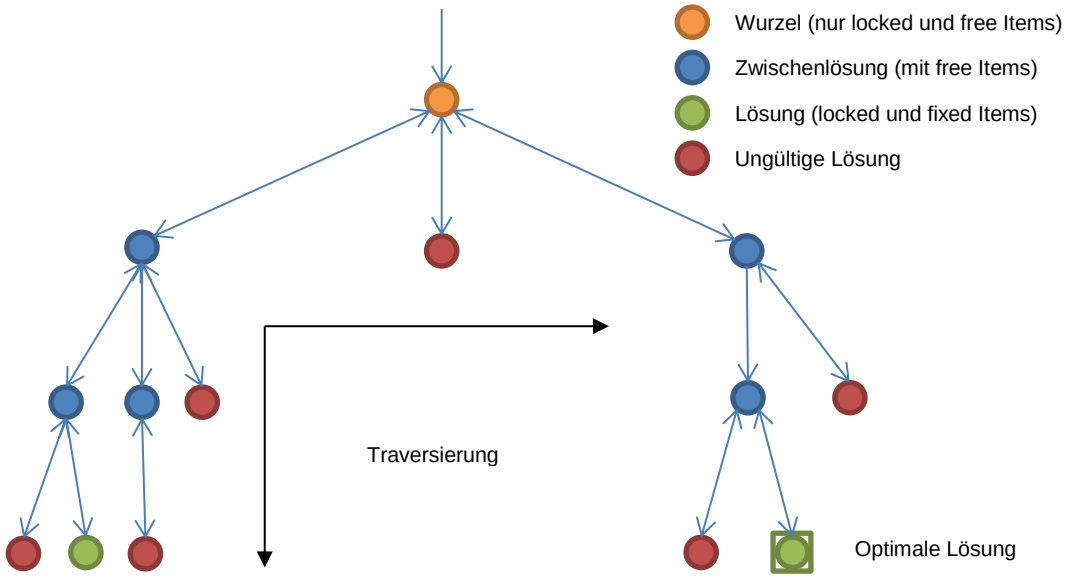


Abbildung 7-40: Such-Baum (Lösungsraum) der Bindung

Beim Binden (bzw. Traversieren) kann zwischen drei Vorgehensweisen gewählt werden:

- Einzige (potentielle) Lösung
  Es findet keine Traversierung statt. Stattdessen wird nur ein einziges Blatt überprüft. Die Ordnung der Aktionen ist hierbei schon vollständig vorgegeben und kann durch die Bindung nicht verändert (Vertauschung) werden.
- Erste (gültige) Lösung
  Die Traversierung stoppt bei der ersten gültigen Lösung.
- Optimale Lösung
  Die Traversierung durchläuft den gesamten Suchbaum. Die letzte gültige Lösung/Blatt ist die optimale Lösung

Wie bereits erläutert wurde, wird das Verhalten beim Abwägen, Binden und Lösen durch die verwendete Strategie bestimmt. Sie wirkt sich somit auf die Hauptfunktionalitäten der Ablaufplanung (reparieren, implementieren und optimieren) aus. Die nachfolgende Tabelle zeigt zusammenfassend das durch die verschiedenen Strategien bestimmte Verhalten des Ablaufplaners.

Strategie und Zweck	**abwägen (Maßnahmen)**	**binden (Traversierung)**	**Verbesserung des Plans notwendig**
umsetzen Entscheidung (normale Priorität)	- Entscheidung (z.B. Hinzufügen Aktivität)	erste Lösung	ja
umsetzen Entscheidung (hohe Priorität) ➔ Schnelligkeit	- Entscheidung (z.B. Hinzufügen Aktivität) - Entfernen von Aktivitäten - Abbruch von Aktivitäten	einzige Lösung	ja
Umsetzen Entscheidung (niedrige Priorität)	- Entscheidung (z.B. Hinzufügen Aktivität) - Alle möglichen Maßnahmen	optimale Lösung	ja
reparieren Plan	- Entfernen von Aktivitäten - Abbruch von Aktivitäten	einzige Lösung	nein

➔ Schnelligkeit			
optimieren Plan	- Alle möglichen Maßnahmen	Optimale Lösung	ja

Tabelle 7-5: Strategien der Ablaufplanung

Zu Beginn des Abschnitts wurde das Zustandsverhalten von Aktivitäten in vereinfachter Form dargestellt (siehe Abbildung 7-30). Diese Sicht erfüllt ihren Zweck, indem sie die elementarsten Zustandsänderungen von Aktivitäten verdeutlicht. Im weiteren Verlauf dieses Abschnitts und des nachfolgenden sollen die Auswirkungen des Ablaufplaners und der Planausführung auf die Zustandssteuerung der Aktivitäten genauer beschrieben werden. Dazu muss das eingeführte Zustandsmodell verfeinert werden. Genauer betrachtet besteht es sogar aus zwei separaten Zustandsmaschinen. Während die erste ihren Schwerpunkt auf die Koordinierung der Ablaufplanung setzt, bestimmt die zweite im Wesentlichen die tatsächliche Ausführung. Obwohl zwischen beiden Zustandsmaschinen gewisse Abhängigkeiten bestehen und sie zum Teil miteinander synchronisiert werden, arbeiten beide größtenteils gleichzeitig und unabhängig voneinander. Dies ist besonders dann der Fall, wenn eine Aktivität durch die Ablaufplanung optimiert werden soll, aber zur selben Zeit durch den *Executor* auch ausgeführt wird.

Der Ablaufplaner koordiniert seine Arbeit mit Hilfe der in Abbildung 7-41 dargestellten Zustandsmaschine für Aktivitäten.

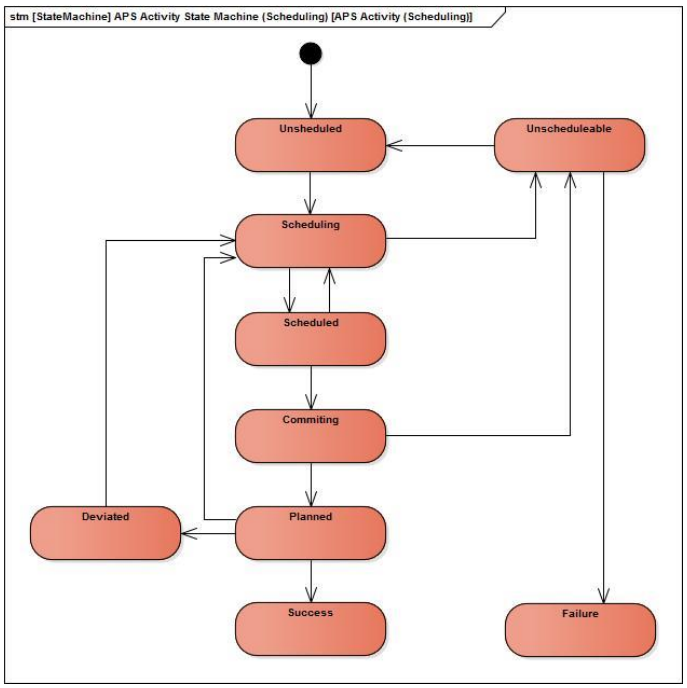


Abbildung 7-41: Zustandsmaschine für Aktivitäten zur Unterstützung der Ablaufplanung

Bei der Umsetzung von Entscheidungen kann der Ablaufplaner neue Aktivitäten erstellen. Diese werden mit Rahmenbedingungen (z.B. frühester Startzeitpunkt) initialisiert, weisen aber zu Beginn noch keine Planung in Form von zugewiesenen Aktionen auf und befinden sich daher im Zustand *Unscheduled*. Der Ablaufplaner versucht daher als nächstes die ungeplanten Aktivitäten in den Zustand *Scheduling* zu versetzen und eine gültige Planung für sie zu finden. Die dafür eingesetzten Methoden wurden bereits im Abschnitt vorgestellt. Wenn eine gültige Lösung für die Aktivität gefunden wurde, geht diese in den Zustand *Scheduled* über, andernfalls erfolgt ein Übergang in *Unscheduleable.* Dieser Zustand besagt lediglich, dass für die Aktivität zur aktuellen Zeit keine gültige Planung gefunden werden kann. Er ist nicht endgültig. Der Ablaufplaner überprüft in regelmäßigen Zeitabständen, ob sich die Rahmenbedingungen für die Aktivität geändert haben und veranlasst ggfs. einen Übergang in *Unscheduled,* um einen erneuten Planungsablauf zu ermöglichen. Aktivitäten im Zustand *Scheduled* sind ebenfalls nicht endgültig, sondern weisen lediglich auf Planänderungen hin, deren Abgleich mit dem tatsächlich auszuführenden Plan noch nicht erfolgt ist. Nachfolgende

Überarbeitungen des Ablaufs der Aktivität verbunden mit einem erneuten Zustandsübergang in *Scheduling* sind weiterhin möglich. Wenn eine gültige Lösung für den gesamten Plan gefunden wurde, können die notwendigen Änderungen übernommen werden. Diese müssen noch, wie bereits erwähnt wurde, durch spezielle Abgleichfunktionen in den aktuellen Plan übertragen werden. Hierzu erfolgt zuerst der Übergang in *Commiting* und bei erfolgreichem Abgleich anschließend in *Planned,* es sei denn, die Änderung hat eine Aktivität aus dem Plan entfernt. In diesem Fall erfolgt der Wechsel in *Unschedulable.* Erst ab diesem Moment, dem erfolgten Abgleich, ist die aktuelle Ablaufplanung der Aktivität im Gesamtplan enthalten. Jede Aktivität wird nach einem erfolgten Abgleich einen besonderen Zähler inkrementieren. Dieser sorgt dafür, dass die Planausführung sämtliche Änderungen mitverfolgen und übernehmen kann. Umgekehrt informiert der *Executor* den Ablaufplaner über alle Abweichungen (mit Hilfe der *Executions*). Wenn solch eine vorliegt, erfolgt für die betroffene Aktivität ein entsprechender Zustandsübergang nach *Deviated.* Nachfolgende Planungsabläufe (über *Scheduling*) können versuchen, diesen Umstand zu beheben. Sobald eine Aktivität auch tatsächlich durch den *Executor* zu Ende ausgeführt wurde, geht diese in den Endzustand *Success* über. Jedoch kann auch der gegenteilige Fall eintreten, dass der Ablaufplaner feststellt, dass für eine Aktivität im Zustand *Unschedulable* eine erfolgreiche Planung generell, auch in Zukunft, ausgeschlossen ist. Dieser Fall trifft besonders dann ein, wenn zeitliche Rahmenbedingungen (z.B. Deadline) nicht mehr eingehalten werden können. Die Folge für die betroffene Aktivität ist ein Übergang in den Endzustand *Failure*.

7.2.5.6 Planausführung

Die Planausführung ist ein Prozess der autonomen Planung, die in einem separaten RODOS-Thread stattfindet. Sie dient der Umsetzung des aktuell gültigen Plans, bestehend aus Aktivitäten, Aktionen, Intervallen und Ressourcentrends. Sie wird durch den Planausführer realisiert und hat zum Ziel, Anweisungen in Form von Kommandos aus dem Plan abzuleiten und an die Steuereinheit des Satelliten, die Kommandoverwaltung, zur Ausführung zu übermitteln. Die nachfolgende Abbildung 7-25 zeigt eine Übersicht der Funktionsweise.

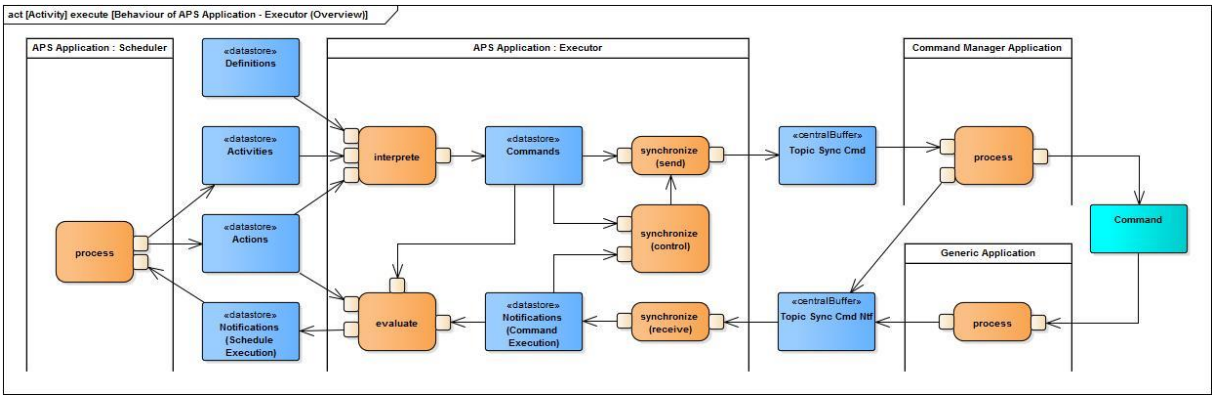


Abbildung 7-42: Interaktionen zur Planausführung (Übersicht)

Die Planausführung ist eine stark datenverarbeitende Interaktion, an der mehrere Funktionalitäten beteiligt sind. Die meisten werden von einer besonderen Komponente, dem Planausführer (*Executor*) bereitgestellt. Eingangsseitig werden im gesamten Ablauf die Bestandteile des Plans, die anweisenden Charakter haben, verwertet. Das sind die Aktionen (*Actions*) und ihre übergeordneten Aktivitäten (*Activities*). Der erste Schritt der Planausführung besteht darin, aus diesen Daten mittels Interpretation (*interprete*) Kommandos (*Commands*) für die Kommandoverwaltung (*Command Manager Application*) zu generieren. Dazu müssen zusätzlich Informationen in Form von Definitionen (*Definitions*) bereitgestellt werden. Sie werden im Rahmen der Konfiguration (siehe Abschnitt 7.2.5.1) eingepflegt und beschreiben im Wesentlichen eine Zuordnung zwischen Aktionen, Kommandos und Kommandoparametern auf abstrakter Ebene. Eine Übersicht der statischen Zusammenhänge erfolgt

an späterer Stelle im Abschnitt. Bei der Interpretation werden konkrete Kommandos erstellt und in eine Liste (im APS) eingepflegt. Der zweite Schritt besteht in einer Synchronisation der Kommandos, da die Kommandoverwaltung ebenfalls eine Liste führt. Da es sich bei der Kommandoverwaltung auch um eine eigenständige RODOS-Anwendung handelt, ist ein Datenaustausch über Anwendungsgrenzen hinweg notwendig. Zu diesem Zweck wurden zwei RODOS-Topics eingerichtet. Auf den ersten Topic, *Sync Cmd,* überträgt der Planausführer die vom Plan abgeleiteten Kommandos. Diese werden kurz vor ihrer Ausführung instanziert und abgeschickt. Die Kommandoverwaltung empfängt die Kommandos und nimmt eine Analyse vor. Dabei müssen die in der nachfolgenden Tabelle strukturierten Daten des Kommandos ausgewertet werden.

Byte-Position	**Name**	**Beschreibung**
0 – 1	Sequence Counter	Der Sequenzzähler des Kommandos. Dient als Schlüssel zur Identifikation und zum Erkennen von Fehlübertragungen.
2 – 3	Command Classifier	Hauptunterscheidungsmerkmal des Kommandos. Dient der Analyse als Hauptkriterium.
4	Command Source	Die Quelle (Operator oder APS), die das Kommando gesendet hat.
5	Command Destination	Der Empfänger (jede beliebige RODOS-Anwendung, auch die Kommandoverwaltung selbst), an den das Kommando gerichtet ist.
6 – 13	Execution Time	Der Zeitpunkt, zu dem das Kommando durch die Kommandoverwaltung ausgeführt werden soll. Dieser hängt vom Wert ab: - =0: Das Kommando wird sofort ausgeführt (immedate command) - <0: Das Kommando wird relativ zum aktuellen Zeitpunkt ausgeführt (time-tagged command) - >0: Das Kommando wird absolut zu einem bestimmten Referenzzeitpunkt ausgeführt (time-tagged command)
14 – 15	Command Index	Der Index des Kommandos. Ermöglicht die Segmentierung des Kommandos.
16 - 17	Parameter Length	Die Länge des angehängten Parameterfelds im Kommando.
18	Flags	Flags, die vorgeben, wie die Kommandoverwaltung und das APS das Kommando zu behandeln haben.
19	Reserved	Ein Reservebyte.
>= 20	Parameter	Die Parameterdaten des Kommandos.

Tabelle 7-6: Der strukturelle Aufbau eines Kommandos

Eine besondere Sicherheitsfunktion nimmt dabei der Sequenzzähler ein. Die Kommandoverwaltung führt mit jeder Kommandoquelle (Operator oder APS) ein separates Zählerpaar. Mit jedem übertragenen Kommando wird der betroffene Zähler inkrementiert. Sollte der Sequenzzähler eines empfangenen Kommandos nicht mit dem erwarteten Wert übereinstimmen, deutet dies auf eine Fehlübertragung hin. Die Kommandoverwaltung wird das empfangene Kommando daraufhin, ohne es zu verarbeiten, verwerfen. Eine Ausnahme von dieser Regelung stellen Kommandos mit dem Zählerwert 0 dar. Diese werden uneingeschränkt von der Kommandoverwaltung akzeptiert, müssen aber sofort (Execution Time = 0) ausgeführt werden. Optional kann, wie im Abschnitt unten beschrieben wird, bei einer erkannten Fehlübertragung eine Rückmeldung gesendet werden, um auf diesen Umstand hinzuweisen.

Zu Synchronisationszwecken kann der Planausführer folgende Anweisungen an die Kommandoverwaltung senden:

- *Get Sequence Counter*: Der Wert des Sequenzzählers der Kommandoverwaltung soll ausgelesen werden.

- *Set Sequence Counter*: Der Wert des Sequenzzählers der Kommandoverwaltung soll überschrieben werden.
- *Clear Command List*: Alle Kommandos in der Kommandoliste sollen gelöscht werden.
- *Append Command:* Ein neues Kommando soll am Ende der Kommandoliste eingefügt werden. Die zeitliche Ordnung darf dabei nicht verletzt werden. Das bedeutet, der Zeitpunkt der Ausführung des einzufügenden Kommandos muss größer als der des letzten Kommandos in der Liste sein.
- *Insert Command:* Ein neues Kommando soll an einer beliebigen Stelle in die Kommandoliste eingefügt werden.
- *Remove Command:* Ein beliebiges bestehendes Kommando soll aus der Kommandoliste entfernt werden.

Die Kommandoverwaltung wird jedes empfangene Kommando, das nicht sofort (immediate) ausgeführt werden soll, in seine eigene Kommandoliste einfügen und dann zum entsprechenden Zeitpunkt ausführen (time-tagged). Kommandos, die für ein anderes Ziel als die Kommandoverwaltung bestimmt sind, werden von dieser über entsprechende Topics zu ihrem Bestimmungsort gesendet. Theoretisch kann dies jede Komponente sein, die eine Anbindung zur RODOS-Middleware besitzt. In der Praxis handelt es sich bei den Komponenten um Anwendungen im Kontext vom APS. Diese werden in der Abbildung 7-42 stellvertretend als generische Anwendungen (*Generic Application*) bezeichnet.

Es gibt zwei Strategien, um Kommandos mitten in eine bereits gefüllte Kommandoliste einzufügen. Die erste sieht vor, die Liste zu löschen (*Clear Command List*) und die alten Inhalte erneut mitsamt des neuen Kommandos schrittweise anzuhängen (*Append Command*). Die zweite verzichtet auf einen kompletten Löschvorgang und fügt das Kommando direkt an der richtigen Stelle ein (*Insert Command*). Das Problem bei der zweiten Variante liegt darin, dass die dafür notwendigen Anweisungen nicht unbedingt von jeder Kommandoverwaltung unterstützt werden. Dies hat historische Gründe, da einzelne Lösch- und Einfügevorgänge möglicherweise die Prozeduren des Satellitenbetriebs korrumpieren könnten, und damit ein zu hohes Risiko darstellen. Es wird daher davon ausgegangen, dass die erste Variante standardmäßig von jeder Kommandoverwaltung unterstützt wird.

Die Übertragung von Kommandos ist äußeren Störungen unterworfen. Dadurch ist ein fehlerfreier Datenaustausch nicht gewährleistet. Für manche Szenarien ist dieser Zustand inakzeptabel, da sie auf eine sichere (d.h. nicht korrumpierende) Übertragung angewiesen sind. Der bereits erwähnte Mechanismus mit dem Sequenzzähler ist unzureichend, da dieser lediglich Fehlübertragungen (z.B. nicht übertragene Kommandos) detektieren kann. Es gibt aber Fälle, bei denen die fehlerfreie Zustellung eines Kommandos unbedingt sichergestellt sein muss. Zu diesem Zweck kann der Planausführer eingehende Rückmeldungen in Form von Mitteilungen (*Notifications - Command Execution*) sowohl von der Kommandoverwaltung als auch von den Zielanwendungen verarbeiten. Diese Mitteilungen werden über das zweite Topic, *Sync Cmd Ntf*, übertragen. Die Verwendung dieser Funktionalität ist optional und kann für jedes Kommando individuell über das bereits oben erwähnte Attribut *Flags* eingestellt werden.

Zu Synchronisierungszwecken stehen insgesamt drei verschiedene Arten von Rückmeldungen zur Verfügung, die je nach Bedarf, auch einzeln oder wahlweise kombiniert, eingesetzt werden können:

- *Reception Notification*: Diese Rückmeldung besagt, ob das betroffene Kommando von der Kommandoverwaltung überprüft, akzeptiert und in die Kommandoliste eingefügt wurde.
- *Invocation Notification*: Diese Rückmeldung besagt, ob das betroffene Kommando von der Kommandoverwaltung aufgerufen und zum Ziel übertragen wurde.
- *Execution Notification*: Diese Rückmeldung besagt, ob das betroffene Kommando vom Ziel ausgeführt wurde.

Die Rückmeldungen liefern Aufschluss über den Erfolg oder Misserfolg der jeweiligen Kommandos in ihren unterschiedlichen Stadien der Abarbeitung. Diese Thematik wird weiter unten im Abschnitt bei der Beschreibung der Zustandsmaschine eines Kommandos aufgegriffen. Die Synchronisation hat auch eine Kontrollfunktion. Sie vergleicht die ausgehenden Kommandos mit den eingehenden Mitteilungen. Dabei können Abweichungen (z.B. negative oder fehlende Bestätigung eines Kommandos) festgestellt werden. Ihnen wird mit passenden Maßnahmen gegengewirkt. Diese können teilweise sehr umfassend sein und schließen in der Regel eine Resynchronisation der Kommandozähler mit ein. Alle Maßnahmen haben zum Ziel, dass die Zustellung eines Kommandos zum Bestimmungsort gewährleistet werden kann.

Des Weiteren muss der Planausführer nicht nur eingehende Mitteilungen verarbeiten, sondern muss auch selbst dem Ablaufplaner (Scheduler) ausgehende Mitteilungen (*Notifications – Schedule Execution*) bereitstellen. Diese Aufgabe erfolgt im Rahmen einer Evaluierung (evaluate). Dabei soll dem Ablaufplaner mitgeteilt werden, ob der Plan wie gewünscht ausgeführt wird. Sollte an dessen Ausführung eine Abweichung vorliegen, müssen Korrekturmaßnahmen ergriffen werden. Diese Aufgabe liegt aber im Verantwortungsbereich des Ablaufplaners. Er ist die Instanz, die auf dieser Ebene (d.h. im Plan) Änderungen vornehmen kann. Der Planausführer kann lediglich bei der Synchronisation, also bei der Übertragung von Kommandos, kontrollierend eingreifen. Die ausgehenden Rückmeldungen beziehen sich daher direkt auf einzelne Aktionen des Plans und können aus den eingehenden Mitteilungen abgeleitet werden. Im Grunde müssen aus diesen und den Kommandos Informationen extrahiert werden, die den Zustand der Ausführung der entsprechenden Aktion beschreiben.

Als nächstes sollen die statischen Zusammenhänge der an der Planausführung beteiligten Komponenten genauer betrachtet werden. Eine Übersicht liefert die folgende Abbildung 7-43.

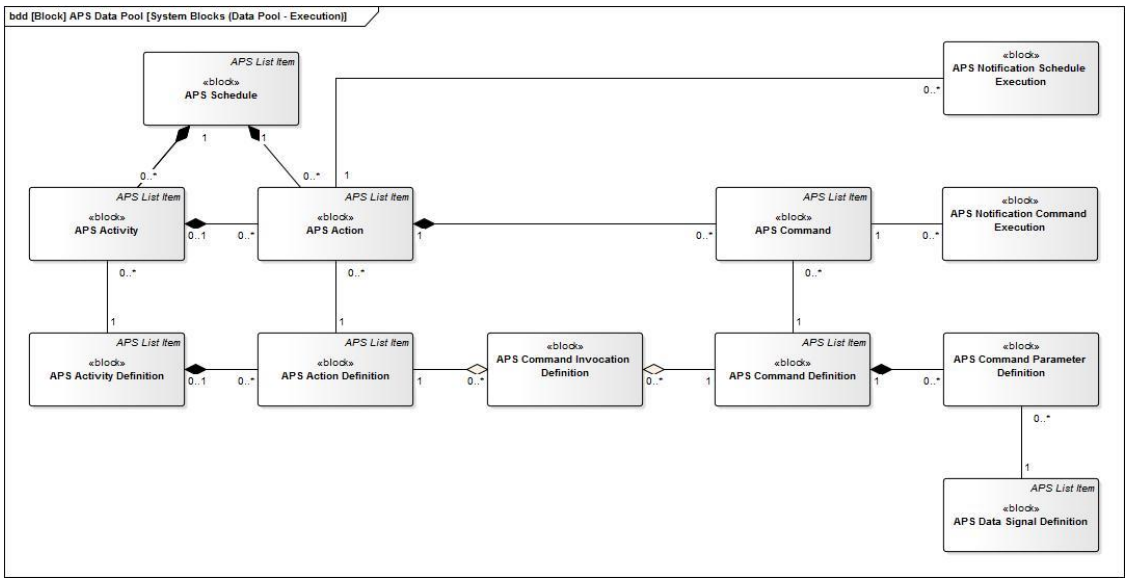


Abbildung 7-43: An der Planausführung beteiligte Blöcke des Daten-Pools

Die Kommandos (*APS Command*) sind die zentralen Gegenstände der Planausführung. Sie werden aus dem aktuellen Plan (APS Schedule) und den darin enthaltenen Aktivitäten (APS Activity) und Aktionen (APS Action) gewonnen. Diese Blöcke wurden schon bei der Beschreibung des Ablaufplaners (siehe 7.2.5.5) vorgestellt. Damit der Planausführer gültige Folgen von Kommandos erstellen kann, benötigt er noch zusätzliches Wissen, das beschreibt, welche Arten von Kommandos generell existieren und wie ihre Inhalte, vor allem die Parameter, aufgebaut sind. Zusätzlich muss noch angegeben werden, wie der Zusammenhang zwischen Aktionen und Kommandos hergestellt wird. Die Verknüpfung wird mittels eines Konstrukts, das einen Kommandoaufruf (Invokation) beschreibt, realisiert. Diese Informationen werden im Rahmen der Konfiguration (siehe Abschnitt 7.2.5.1) eingelesen. Im Grunde handelt es sich um Definitionen (*APS * Definition*), die ein Szenario

beschreiben. Dementsprechend existieren Definitionen für Aktivitäten (*APS Activity Definition*), Aktionen (*APS Action Definition*), Kommandos (*APS Command*), Kommandoparameter (*APS Command Parameter Definition*) und Kommandoaufrufen (*APS Command Invocation Definition*).

Während der Ablaufplanung wurde für jede Aktion ein Ausführungszeitpunkt festgelegt. Der Planausführer wird eine bestimmte Zeitspanne vor dem gesagten Zeitpunkt mit der Instanzierung der zur Aktion referenzierten Kommandos beginnen. Der Instanzierungshorizont ist der auf die aktuelle Zeit bezogene Zeitbereich, in dem die Instanzierung der Kommandos eingeleitet wird. Er ist relativ klein, so dass von einem „Just-In-Time“ - Verhalten auszugehen ist. Dabei ermittelt die Planausführung über die Definitionen, insbesondere über die Aufrufe (*APS Command Invocation Definition*), welche Kommandos für die Aktion zu erstellen sind. Hierbei müssen die relativen Zeitangaben der Aufrufe in absolute umgerechnet werden. Die Parameter der jeweiligen Kommandos werden bei deren Instanzierung zuerst mit Standardwerten versehen. Diese können im Bedarfsfall in einem weiteren Schritt mit Hilfe der Parameterdefinition wahlweise mit Werten der Datensignale überschrieben werden. Damit können die erstellten Kommandos flexibel angepasst werden. Sie werden zentral in einer Liste verwaltet. Diese ist primär nach den Ausführungszeitpunkten geordnet. Trotz dieser Ordnung kann es vorkommen, dass sich die Ausführungszeiträume der Kommandos zweier oder mehrerer Aktionen überlappen. Es ist auch zu beachten, dass bei der Instanzierung der Kommandos der jeweilige Sequenzzähler noch nicht gesetzt wird. Dies erfolgt erst, wenn ein weiterer Zeithorizont, der Synchronisierungshorizont, den Ausführungszeitpunkt des jeweiligen Kommandos erfasst. Dadurch kann es vorkommen, dass die Sequenzzähler und die Ausführungszeitpunkte der Kommandos eine unterschiedliche Ordnung besitzen. Die zwei zuvor im Abschnitt erwähnten Synchronisierungsstrategien lösen dieses Problem auf unterschiedliche Weise. Die erste Strategie löscht bei Bedarf die komplette Kommandoliste und baut diese im Anschluss mit korrekter Ordnung neu auf. Die zweite Strategie manipuliert die Kommandoliste gezielt an einzelnen Stellen. Infolgedessen muss sie durch die Kommandoverwaltung evtl. neu sortiert werden. Dieser Fall tritt besonders dann ein, wenn der Plan kurzfristig geändert wurde, um eine hochpriorisierte Entscheidung, die auch unmittelbar ausgeführt werden soll, umzusetzen.

Die Synchronisation soll anhand eines Beispiels veranschaulicht werden. In den nachfolgenden drei Tabellen sind die dafür verwendeten konkreten Werte gelistet.

Es werden drei Aktionen definiert, denen jeweils zwei Kommandos zugeordnet sind:

Name der Aktion (Definition)	**Name des Kommandos (Definition)**	**Ausführungszeitpunkt (relativ)**
AD1		
	AD1.CD1	1
	AD1.CD2	20
AD2		
	AD2.CD1	0
	AD2.CD2	1
AD3		
	AD3.CD1	3
	AD3.CD2	5

Tabelle 7-7: Definition (Beispiel)

Der umzusetzende Plan soll folgende Aktionen aufweisen:

Name der Aktion (Plan)	**Name der Aktion (Definition)**	**Ausführungszeitpunkt (absolut)**
Aktion 1	AD1	8
Aktion 2	AD2	15
Aktion 3	AD3	19

Tabelle 7-8: Plan (Beispiel)

Die Horizonte weisen folgende Werte auf:

Horizont	Dauer
Instanzierungs-Horizont	12
Synchronisierungs-Horizont	6

Tabelle 7-9: Horizonte (Beispiel)

Abbildung 7-44 zeigt den zeitlichen Verlauf der Horizonte und gibt die verschiedenen Zeitpunkte der Planerstellung im Beispiel an:

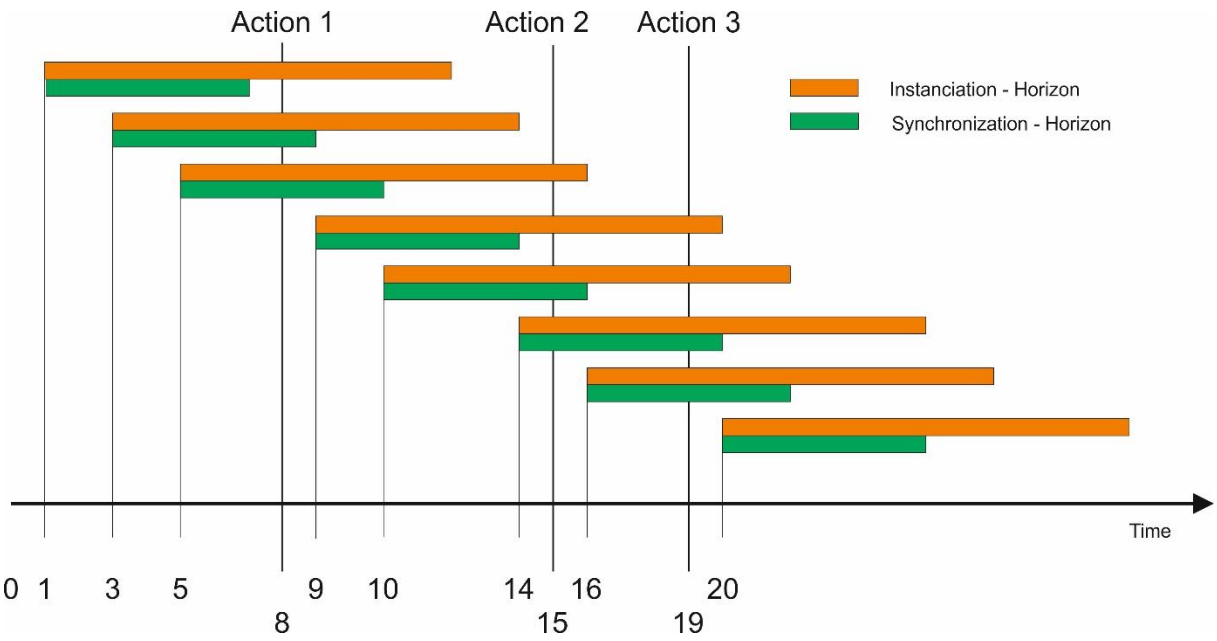


Abbildung 7-44: Darstellung der Horizonte und der zeitlichen Referenzpunkte (Beispiel)

Die nachfolgende Tabelle zeigt die im Beispiel durch den Planausführer erstellten Kommandos in Abhängigkeit der genannten Referenzzeitpunkte. Mit dem Voranschreiten der Zeit durchläuft jedes Kommando der Reihe nach mehrere Zustände:

- Bereit: Das Kommando wurde instanziert, aber noch nicht mit der Kommandoverwaltung synchronisiert
- Übertragen: Das Kommando wurde im Rahmen der Synchronisation mit der Kommandoverwaltung übertragen, aber noch nicht ausgeführt
- Ausgeführt: Das Kommando wurde durch die Kommandoverwaltung ausgeführt.

Der Planausführer koordiniert seine Arbeit mit Hilfe dreier Zustandsmaschinen, je eine für Aktivitäten, Aktionen und Kommandos. Diese stehen gegenseitig in Wechselwirkung. Sie sollen im Folgenden genauer beschrieben werden. Auf untersteter Ebene steht die Zustandsmaschine für die Kommandos (siehe Abbildung 7-45).

Aktueller Zeitpunkt (absolut)	Kommandos	Sequenzzähler	Zeitpunkt der Ausführung (absolut)	Zustand
1	A1.C1	1	9	Bereit
	A1.C2	2	28	Bereit
3	A1.C1	1	9	Übertragen
	A1.C2	2	28	Bereit
5	A1.C1	1	9	Übertragen
	A2.C1	3	15	Bereit
	A2.C2	4	16	Bereit
	A1.C2	2	28	Bereit
9	A1.C1	1	9	Ausgeführt
	A2.C1	3	15	Übertragen
	A2.C2	4	16	Bereit
	A3.C1	5	22	Bereit
	A3.C2	6	24	Bereit
	A1.C2	2	28	Bereit
10	A1.C1	1	9	Ausgeführt
	A2.C1	3	15	Übertragen
	A2.C2	4	16	Übertragen
	A3.C1	5	22	Bereit
	A3.C2	6	24	Bereit
	A1.C2	2	28	Bereit
14	A1.C1	1	9	Ausgeführt
	A2.C1	3	15	Übertragen
	A2.C2	4	16	Übertragen
	A3.C1	5	22	Bereit
	A3.C2	6	24	Bereit
	A1.C2	2	28	Bereit
16	A1.C1	1	9	Ausgeführt
	A2.C1	3	15	Ausgeführt
	A2.C2	4	16	Ausgeführt
	A3.C1	5	22	Übertragen
	A3.C2	6	24	Bereit
	A1.C2	2	28	Bereit
20	A1.C1	1	9	Ausgeführt
	A2.C1	3	15	Ausgeführt
	A2.C2	4	16	Ausgeführt
	A3.C1	5	22	Übertragen
	A3.C2	6	24	Übertragen
	A1.C2	2	28	Bereit

Tabelle 7-10: Erstellte Kommandofolge in Abhängigkeit der Zeit (Beispiel)

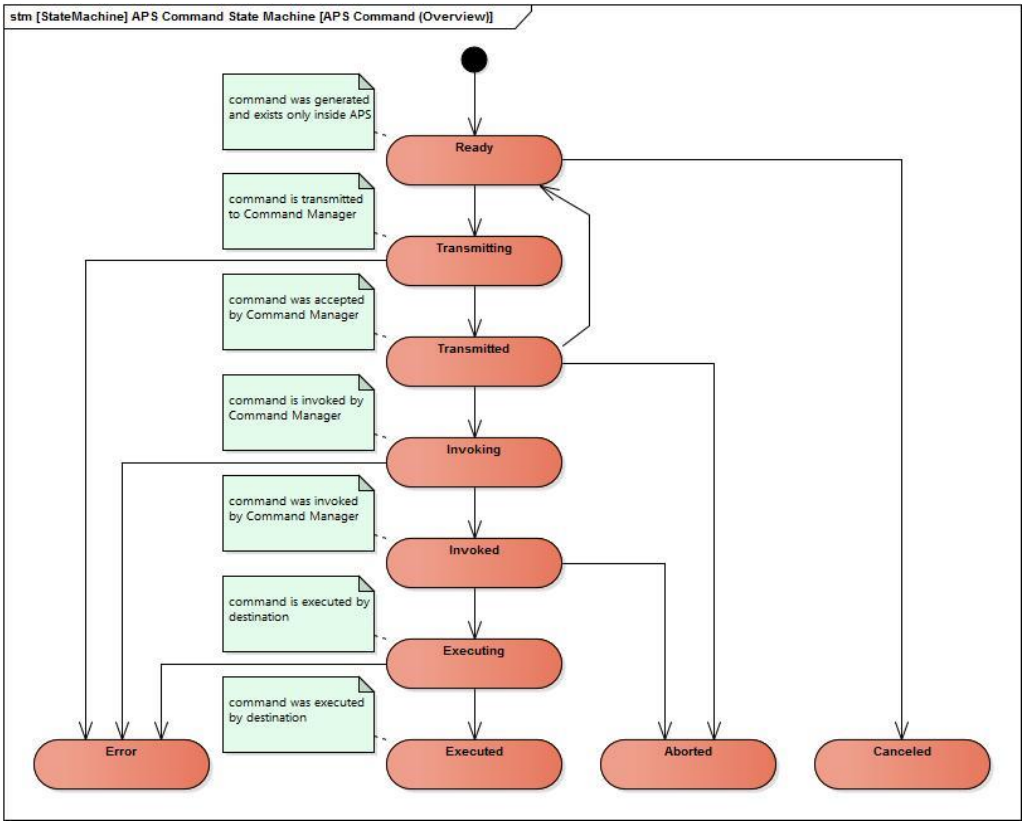


Abbildung 7-45: Zustandsmaschine für Kommandos zur Unterstützung der Planausführung

Ein vereinfachtes Modell der Zustandsmaschine wurde bereits bei der Funktionsbeschreibung des Planausführers anhand des Beispiels eingeführt. Es wird nun auf den vollen Funktionsumfang erweitert. Ein Kommando beginnt seinen Lebenszyklus mit der Instanzierung im Zustand *Ready*. In diesem verweilt es so lange, bis es vom Planausführer im Rahmen der Synchronisierung an die Kommandoverwaltung übertragen werden soll. Wenn dieser Fall eintritt, bekommt das Kommando eine Sequenznummer zugewiesen und wechselt in den Zustand *Transmitting*. Die Verweildauer in diesem ist relativ kurz und soll nur zeigen, dass das Kommando derzeit übertragen wird. Nach Abschluss der Übertragung erfolgt ein Wechsel in *Transmitted*. Wenn bei der Synchronisierung keine Rückmeldungen (*Notifications*) ausgewertet werden sollen (kann über die Flags des Kommandos eingestellt werden), erfolgt dieser Übergang automatisch, andernfalls hängt der weitere Verlauf von der Auswertung der Rückmeldung ab. Bei Unstimmigkeiten (negative Rückmeldungen) wechselt das Kommando in den Zustand *Error. Transmitted* bedeutet auch, dass das Kommando in die Liste der Kommandoverwaltung eingepflegt wurde und auf seine Abarbeitung wartet. Es kann aber auch passieren, dass das Kommando nach einiger Zeit von der Liste der Kommandoverwaltung entfernt werden soll. In diesem Fall erfolgt ein Wechsel zurück in *Ready*. Die Schleife kann dann erneut durchlaufen werden. Dabei wird jedoch wieder eine neue Sequenznummer zugewiesen. Im weiteren Verlauf wiederholt sich das soeben beschriebene Prinzip der Zustandsmaschine. Erneut ist eine Übertragung des Kommandos notwendig. Dieses Mal allerdings wird der Vorgang von der Kommandoverwaltung selbstständig angestoßen. Diese überträgt das Kommando zur ausführenden Komponente. Der Planausführer wird über diesen Vorgang lediglich informiert. In der Zustandsmaschine werden diese Wechsel mit den Zuständen *Invoking* (für Dauer der Übertragung) und *Invoked* abgebildet. Letzterer besagt, dass das Kommando seitens der Kommandoverwaltung ausgeführt und zum Bestimmungsort übertragen wurde. Wie zuvor beschrieben wurde, können auch hierbei Probleme auftreten, sodass ebenfalls ein Wechsel in den Zustand *Error* möglich ist. Zum Schluss muss das Kommando von der Zielkomponente auch tatsächlich ausgeführt werden. Dieses Verhalten wird mit den Zuständen *Executing* und *Executed* abgebildet. Der letztgenannte Zustand bedeutet, dass das Kommando seitens der ausführenden Komponente erfolgreich abgearbeitet wurde und somit auch den Endzustand darstellt.

Wie bereits erwähnt wurde, können bei der Abarbeitung eines Kommandos Fehler (durch negative Rückmeldungen) auftreten und den Wechsel in den Zustand *Error* erzwingen. Dieser besagt, dass das Kommando auf unerwarteter Weise nicht abgearbeitet werden kann. Es kann aber auch der Fall eintreten, dass die Abarbeitung eines Kommandos bewusst, d.h. durch Planung, abgebrochen werden soll. In diesem Fall liegt kein Fehlverhalten vor. Um diesen Übergang zu ermöglichen, wurden noch zwei weitere Endzustände, die einem Abbruch entsprechen, in die Zustandsmaschine aufgenommen. Zum einem kann ein Kommando annulliert und in den Zustand *Canceled* versetzt werden. Das bedeutet, es hat auf der Seite der Kommandoverwaltung nie existiert und kann dementsprechend nur vom Zustand *Ready* aus überwechseln. Zum anderen muss der Fall berücksichtigt werden, dass mit dem Kommando bereits eine Synchronisation mit der Kommandoverwaltung erfolgt ist. Dadurch bedingt existiert das Kommando bereits auf der Seite der Kommandoverwaltung und muss durch spezielle Synchronisierungsvorgänge gelöscht werden. Dieser Vorgang stellt einen massiven Eingriff in die Liste der Kommandoverwaltung dar. Er wird entsprechend den beschriebenen Synchronisierungsstrategien gehandhabt. In Abgrenzung zum vorherigen Abbruchvorgang erfolgt der Zustandsübergang hierbei in *Aborted* und kann nur von *Transmitted* und *Invoked* erfolgen.

Das Synchronisierungszeitfenster zwischen Planausführer und Kommandoverwaltung legt fest, wann die Übertragung eines Kommandos zwischen diesen beiden ausführenden Komponenten eingeleitet wird. Wenn das Zeitfenster hinreichend klein gewählt wird – dies ist zur Vermeidung unnötiger Plankorrekturen auch vorgesehen – kann die autonome Kommandierung auch als „Just-In-Time“, mit quasi „immediate“ Kommandos (tatsächlich aber mit einem ganz kleinen Time-Tag), aufgefasst werden. Dass der Planausführer eine Kommandoliste pflegt, ist für die Kommandoverwaltung transparent.

Als nächstes soll die Zustandsmaschine für die Aktionen (siehe Abbildung 7-46) betrachtet werden. Sie steht in Wechselwirkung mit den Zustandsmaschinen der einzelnen Kommandos, die der jeweiligen Aktion zugeordnet sind. Ihre Zustände bestehen aus Hauptzuständen und Unterzuständen.

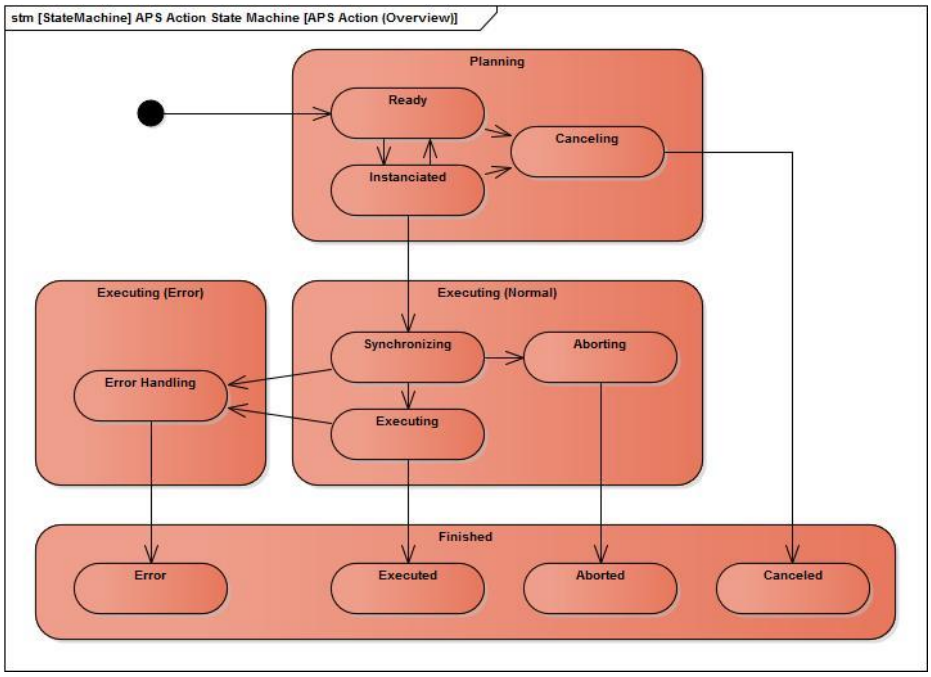


Abbildung 7-46: Zustandsmaschine für Aktionen zur Unterstützung der Planausführung

Eine Aktion beginnt ihren Lebenszyklus im Zustand *Planning/Ready,* nachdem sie vom Ablaufplaner erstellt wurde. Der Hauptzustand besagt, dass die Eigenschaften einer Aktion ohne Probleme geändert werden können, da weder die Kommandoverwaltung noch die Zielkomponente durch die Synchronisierung dahingehend beeinflusst wurden. In der Regel wird der Ablaufplaner einige Änderungen (z.B. durch Optimierung) an der Aktion vornehmen. Erst mit dem Eintreten der Aktion in

den Instanzierungshorizont wird der Planausführer aktiv. Wie bereits im Abschnitt beschrieben, werden dann Kommandos von der Aktion abgeleitet. Die Folge ist ein Übergang des Zustandes in *Planning/Instanciated*. Zu diesem Zeitpunkt sind Änderungen an der Aktion immer noch unproblematisch, da die zugehörigen Kommandos nur im Planausführer existieren. Diese müssen ggfs. entsprechend angepasst bzw. gelöscht werden.

Im weiteren Verlauf tritt irgendwann der Zeitpunkt ein, zu dem die Aktion vom Synchronisierungshorizont erfasst wird. Dies hat einen Wechsel in den Zustand *Executing(Nominal)Synchronizing* zur Folge. Ab diesem Zeitpunkt sind Änderungen an der Aktion nicht mehr zulässig, da ihre Synchronisierung und ihre anschließende Ausführung, d.h. die der zugeordneten Kommandos, unmittelbar bevorstehen. Diese Maßnahme (Sperrung) soll Inkonsistenzen zwischen dem Planausführer und der Kommandoverwaltung bei der Ausführung der Aktionen verhindern. Mit dem Aufruf des ersten Kommandos, d.h. wenn dieser in den Zustand *Executing* (siehe Abbildung 7-45) eintritt, erfolgt für die Aktion ein Zustandswechsel in *Executing(Nominal)/Executing*. Dieser wird solange beibehalten, bis das letzte Kommando erfolgreich ausgeführt wurde. Anschließend findet ein letzter Wechsel in den Endzustand *Finished/Executed* statt. Die soeben beschriebenen Übergänge, von der Planung (*Planning*) bis zum Abschluss (*Finished*), zeigen den erwarteten nominalen Verlauf. Von diesem weicht der Planausführer unter Umständen ab, insbesondere dann, wenn Fehlverhalten und Abbrüche behandelt werden sollen. Ersteres triff auf, wenn eines der Aktion zugeordneten Kommandos nicht ordnungsgemäß, sondern mit einem Fehler, abschließt. In diesem Fall wird eine Fehlerbehandlung, verbunden mit einem Übergang in den Zustand *Execution(Error)/Error Handling,* gestartet. In Folge werden alle restlichen zur Aktion gehörenden Kommandos abgebrochen. Im Anschluss wechselt die Aktion in den Endzustand *Finished/Error*. Die zuletzt genannte Abweichung - die Abbrüche - wird genauso wie bei der Zustandsmaschine für die Kommandos beschrieben, bewusst durchgeführt und gilt daher nicht als Fehlverhalten. Auch hier kann eine Aktion entweder direkt annulliert (über *Canceling*) werden, wenn sie noch nicht mit der Kommandoverwaltung synchronisiert wurde, ansonsten muss ein harter Abbruch (über *Aborting*) durchgeführt werden. Unabhängig von der Art des Abbruchs, hat dieser zur Folge, dass sämtliche zugeordnete Kommandos der Aktion ebenfalls abgebrochen werden.

Zu Beginn der Beschreibung des Ablaufplaners (siehe Abschnitt 7.2.5.5) wurde das Zustandsverhalten von Aktivitäten in vereinfachter Form dargestellt (siehe Abbildung 7-30). Im Nachhinein betrachtet, wurde darin der Aspekt der Ausführung modelliert. Diese Thematik wird an dieser Stelle wieder aufgegriffen und erweitert, sodass sie mit den bisher gezeigten Planabläufen bezüglich Aktionen und Kommandos kompatibel ist. Das Ergebnis ist eine Zustandsmaschine für Aktivitäten wie sie in Abbildung 7-47 dargestellt ist.

Die Zustandsmaschine der Aktivitäten weist einige Gemeinsamkeiten mit derjenigen der Aktionen auf. Sie besteht ebenfalls aus Haupt- und Unterzuständen, die die einzelnen Aspekte der Planung (*Planning*), der Ausführung (*Executing Nominal*), der Fertigstellung (*Finished*) und der Fehlerbehandlung (*Executing Error*) abbilden. Der Lebenszyklus einer Aktivität beginnt im Zustand *Planning/Unscheduled,* nachdem sie vom Ablaufplaner instanziert wurde. Dieser wird im Anschluss versuchen, mit den im Abschnitt 7.2.5.5 beschriebenen Methoden eine gültige Planung für die Aktivität zu finden und entsprechende Aktionen zu berechnen. Bei Erfolg erfolgt ein Wechsel in den Zustand *Planning/Ready*.

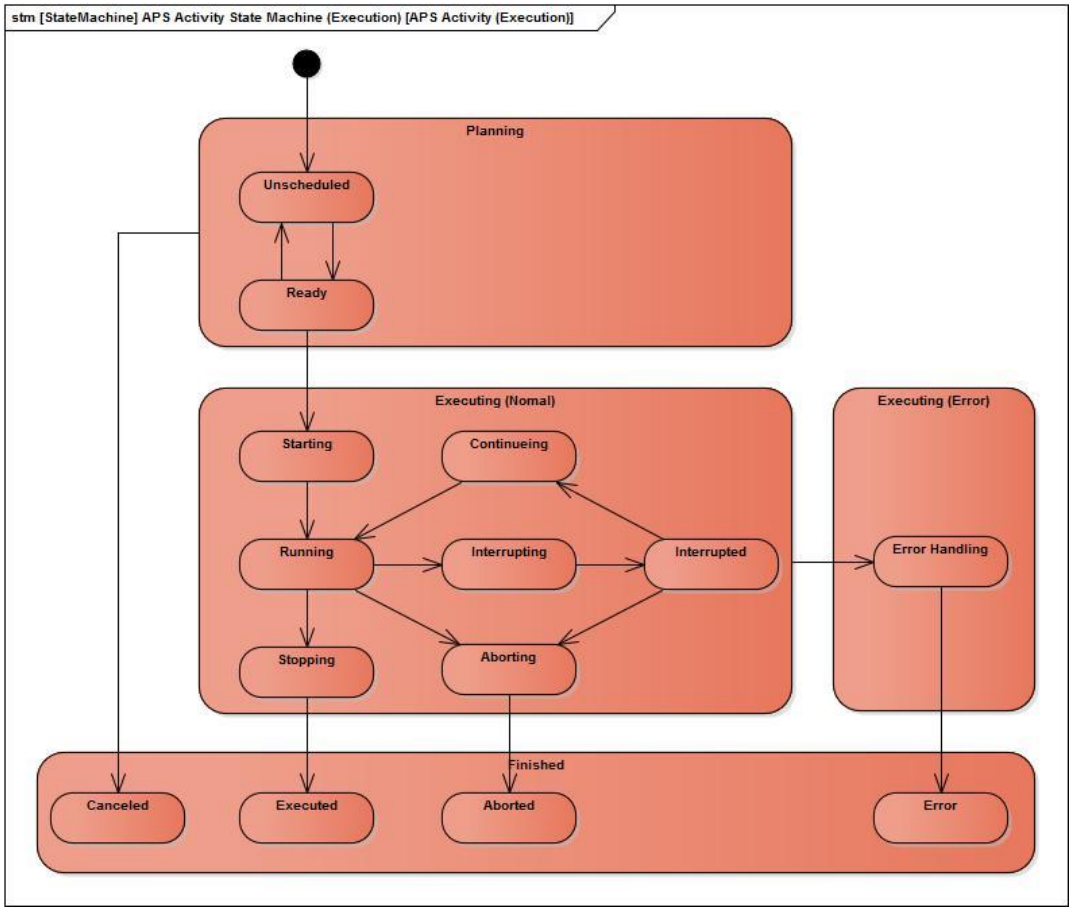


Abbildung 7-47: Zustandsmaschine für Aktivitäten zur Unterstützung der Planausführung

Dieser Übergang kann unter Umständen in Rahmen der Planung wieder umgekehrt werden. Auf die bisher aufgeführten Schritte hat der Planausführer keinen Einfluss, da diese nicht in seinem Verantwortungsbereich liegen. Erst wenn die erste Aktion der Aktivität, eine Startaktion, zeitlich gesehen vom Synchronisierungshorizont erfasst wird, greift der Planausführer ein. Die Aktivität wechselt dann in den Hauptzustand *Executing(Nominal)*. Die Verknüpfungen der sich darin befindenden Unterzustände bilden im Grunde das zuvor erwähnte vereinfachte Modell der Aktivitätenzustandsmaschine mit den Zuständen Laufend (*Running*) und Angehalten (*Interrupted*). Die Wechsel zu und von diesen werden bekanntermaßen durch die Aktionen (Start, Stop, Interrupt, Continue, Abort) eingeleitet. Ihre Abarbeitung wird jeweils in einem eigenen Unterzustand (*Starting, Stopping, Interrupting, Continueing*) gekapselt und dient dazu, dass die entsprechenden untergeordneten Aktionszustandsmaschinen abgearbeitet werden können. Mit einer Stoppaktion wird eine Aktivität beendet, sodass sie in den Endzustand *Finished/Executed* wechselt. Die soeben beschriebenen Übergänge, von der Planung (*Planning*) bis zum Abschluss (*Finished*), zeigen, wie schon bei der Aktionszustandsmaschine, den erwarteten nominalen Verlauf. Auch hiervon kann es wieder zwei verschiedene Arten von Abweichungen geben, die ähnlich behandelt werden. Fehler werden im gesonderten Zustand *Executing(Error)/Error Handling* behandelt und schließen im Endzustand *Finished/Error* ab. Abbrüche hingegen werden durch eine eigene Aktion behandelt und schließen im Endzustand *Finished/Aborted* ab.

7.2.6 Besondere Aspekte

Dieser Abschnitt soll zum Schluss noch auf besondere Aspekte, die es bei der Anwendung des APS zu beachten gilt, eingehen.

7.2.6.1 Softwaredienste

Das Verhalten des APS hängt von seiner Konfiguration und den Stimuli ab. Beide Aspekte sind durch die Referenzszenarien vorgegeben. Zu den Stimuli gehören in der Regel die synchronisierten Zustandsdaten (Housekeeping) des Satelliten und die Meta – Daten. Letztere werden durch entsprechende Quellen, die im Kontext des APS angesiedelt sind, bereitgestellt. Bei diesen Quellen handelt es sich in der Regel um spezielle Softwareanwendungen, auch Softwaredienste genannt, die einen formalen Aufbau (siehe Abbildung 7-48) haben.

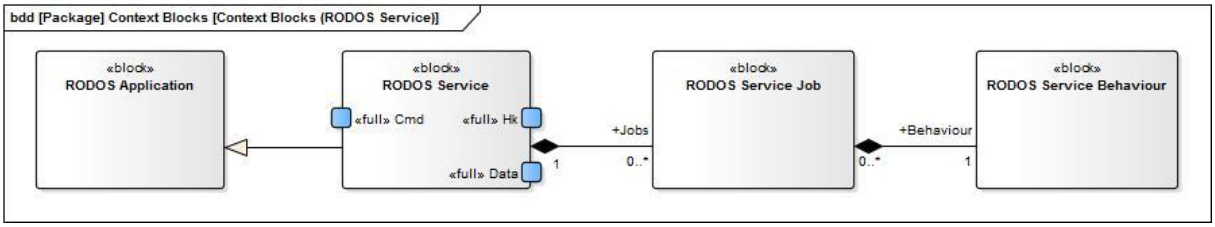


Abbildung 7-48: struktureller Aufbau von Softwarediensten

Ein Softwaredienst (*RODOS Service*) ist eine spezialisierte Softwareanwendung (*RODOS Application*). Sie verfügt daher ebenfalls wie die allgemeine Anwendung über zwei Standard-Topics, den einen zur Kommandierung (Topic *Cmd*) und den anderen zur Telemetrierung (Topic *Hk*). Zusätzlich verfügt sie über einen weiteren Middlewarezugang, über den sie ihre Ergebnisse (Topic *Data*) ausgeben kann. Das APS zum Beispiel nutzt diese Möglichkeit, um über diesen Mechanismus Meta-Daten zu beziehen. Ein Softwaredienst hat eine oder mehrere Aufgaben (*RODOS Service Job*), die mittels Kommandierung eingestellt werden können. Zu jeder Aufgabe lassen sich einige Eigenschaften, die alle Softwaredienste gemeinsam haben, festlegen:

- Zyklische Ausführung (Ja/Nein)
- Zyklusdauer (falls zyklisch)
- Zeitpunkt der nächsten Ausführung
- Anzahl der verbliebenen Ausführungen (falls zyklisch)
- Anzahl der erfolgten Ausführungen (falls zyklisch)
- Das RODOS-Topic, über das die Ergebnisse als Botschaft ausgegeben werden sollen

Die eigentlichen Funktionalitäten, durch die sich die verschiedenen Dienste unterscheiden, sind in separaten Verhaltenskomponenten (*RODOS Service Behaviour*) gekapselt. Der soeben beschriebene strukturelle Aufbau von Softwarediensten ist nur als Grundgerüst zu betrachten. Konkrete Implementierungen eines Dienstes, z.B. zur Berechnung von Orbit-Informationen (siehe Abschnitt 7.3.5.3.3), werden durch spezialisierte Verhaltensweisen (abgeleitet von *RODOS Service Behaviour*) realisiert.

7.2.6.2 Sicherheit des Raumsegments

Beim Entwurf eines Raumsegments wird der Sicherheit sehr viel Bedeutung beigemessen. Die Subsysteme und Komponenten werden so konzipiert, dass nach Möglichkeit keine Fehler entstehen, da diese unter Umständen zu einem Totalausfall der ganzen Raumplattform führen können.

Ein autonomes Planungssystem stellt eine erhebliche Gefahrenquelle dar. Dies hat mehrere Gründe:

- Autonome Planungssysteme greifen aktiv in die Steuerung des Satelliten ein und können durch falsche Konfiguration erheblichen Schaden anrichten.
- Autonome Planungssysteme befinden sich derzeit in der Erforschung und wurden bisher nur in wenigen Satellitenmissionen eingesetzt. Wegen der fehlenden Erfahrung können vermehrt unvorhersehbare kritische Situationen eintreten.

Um einen Satelliten generell vor Fehlern und Störungen zu schützen, wird dieser mit einer Reihe von Sicherheitsmaßnahmen ausgestattet. In der Systementwicklung hat sich dazu der Begriff FDIR (failure detection, isolation and recovery) etabliert. Darunter werden folgende Funktionsweisen zusammengefasst:

- Erkennung von eingetretenen Fehlern (*failure detection*)
- Die Isolation von Fehlern, sodass diese nach Möglichkeit das Gesamtsystem nicht weiter gefährden (*isolation*)
- Die Wiederherstellung des Normalzustandes nach dem Eintritt eines Fehlers (*recovery*)

In der Raumfahrttechnik hat sich folgende, in Abbildung 7-49 dargestellte, Vorgehensweise etabliert.

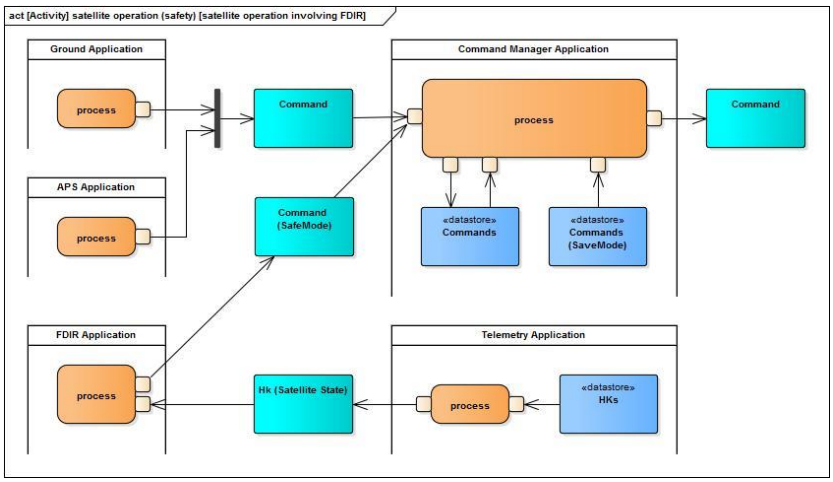


Abbildung 7-49: Einbindung der FDIR-Anwendung in den Planungsprozess

Die Verantwortung für die FDIR-Funktionalität trägt eine dedizierte sich im Kontext des APS befindende Anwendung (*FDIR Application*). Diese kann den aktuellen Zustand (*HK Satellite State*) des Satelliten mit Hilfe der Telemetrie-Anwendung einsehen. Diagnose-Algorithmen der FDIR-Anwendung untersuchen diese Daten auf Auffälligkeiten. Dabei festgestellte Fehler am Satelliten oder an einer seiner Komponenten (failure dectection) gilt es zu isolieren (*isolate*). Dazu wird die FDIR-Anwendung die Kommando-Anwendung des Satelliten mittels eines speziellen Kommandos (*Command Safe Mode*) auffordern, den aktuellen Betrieb, sei er konventionell oder autonom, einzustellen, und in den sogenannten „Safe Mode“ überzugehen. Die Kommando-Anwendung wird dazu eine besondere Sequenz an Kommandos ausführen, die in einer separaten Liste (*Commands Safe Mode*) gespeichert sind. Solch eine Sequenz wird in der Regel folgenden Aufbau haben:

1. Unterbinde die Ausführung von Kommandos in der Kommandoliste
2. Deaktiviere sofort (d.h. *safe mode commands* als *immediate commands*) sämtliche Komponenten (auch das autonome Planungssystem) des Satelliten, die für den Betrieb im Safe-Mode nicht benötigt werden.
3. Lösche den Inhalt der nominalen Kommandoliste (*Commands*).
4. Versetze die Lageregelung des Satelliten in einen Modus, der dazu geeignet ist, den Energiespeicher (Batterie) des Satelliten zu erhalten. In der Regel wird der Satellit eine Lage einnehmen, sodass die Solarpaneele zur Sonne hin ausgerichtet sind.

Für die Wiederherstellung (*recovery*) ist der Operator zuständig. Dieser wird in der Regel geeignete Maßnahmen treffen, um diesen Fehlerzustand in Zukunft zu vermeiden. Zum Beispiel kann er eine fehlerhafte Komponente im Satelliten als fehlerhaft maskieren, und so deren Verwendung für die Zukunft ausschließen. Zum Schluss wird der Operator ein Kommando senden, um den Satelliten wieder zurück in den nominalen Betrieb zu versetzen.

Das APS wird bzgl. der FDIR-Anwendung wie eine gewöhnliche Komponente/Anwendung behandelt. Es stellt eventuell (durch falsche Konfiguration) ein erhöhtes Fehlerrisiko dar, wird aber im Rahmen der Fehlerbehandlung genauso wie jede andere gewöhnliche Komponente isoliert. Des Weiteren besitzt das APS teilweise Funktionalitäten zum Erkennen (failure detection) von Ausnahmezuständen. Bei der Erklärung der Zustandsmaschinen von Aktivitäten und Aktionen wurden auch deren Fehlerzustände erwähnt. Der Übergang in solch einen Fehlerzustand entspricht schon der Fehlererkennung. Der Operator kann im Rahmen der Konfiguration des APSs die Art der Fehlerbehebung festlegen. Entweder versucht es eigenständig den Fehler durch einen entsprechenden Abbruch der Aktivität zu isolieren, oder es löst bei der Abarbeitung der Abbruch-Aktion einen Übergang in den Safe-Mode aus. Dies geschieht durch das Verschicken des entsprechenden Safe-Mode-Kommandos, das mit der Abbruch-Aktion per Definition verknüpft ist. Die Fehlerhandhabung des APS soll auf keinen Fall dazu verwendet werden, die FDIR-Anwendung zu ersetzen, sondern ist lediglich als eine Ergänzung im Sicherheitskonzept des Satelliten gedacht.

7.2.6.3 Kommunikation mit Operator

In den bisherigen Abschnitten wurde meist nur die Kommunikation des APSs mit seinen unmittelbaren Nachbarn, insbesondere den Anwendungen zur Kommandoverwaltung und zur Zustandsdaten-erfassung, im Kontext betrachtet. Der Datenaustausch dient meistens den Synchronisations-vorgängen der autonomen Planung. Es gibt jedoch Kommunikationspfade, in die auch der Operator am Boden involviert ist (siehe Abbildung 7-50).

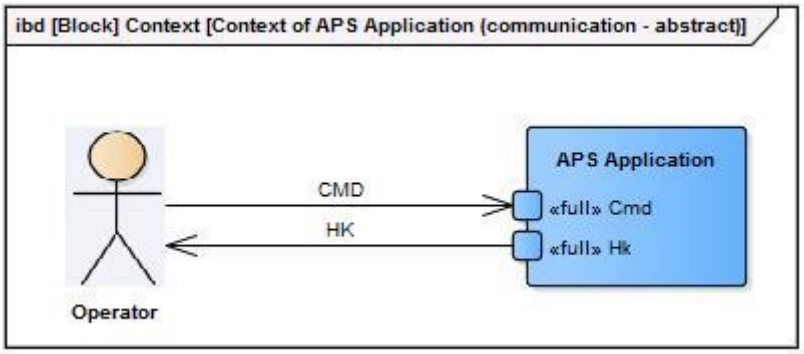


Abbildung 7-50: Kommunikation zwischen Operator und APS (abstrakte Darstellung)

Aus Sicht des Operators und des APSs ist die Funktionsweise der Kommunikationspfade irrelevant. Deswegen wurde diese bisher auch nicht beachtet. Das APS ist eine RODOS-Anwendung. Es empfängt seine Kommandos und verschickt seine Telemetrie über RODOS-Topics. Die untergeordneten Abläufe der Kommunikation bleiben für die Anwendung verborgen.

Zum besseren Verständnis sollen die Kommunikationspfade an dieser Stelle dennoch im Detail betrachtet werden (siehe Abbildung 7-51). Die Vorgänge werden exemplarisch am Pfad für die Kommandierung erklärt. Der entgegengesetzte Pfad zur Telemetrierung ist analog in umgekehrter Reihenfolge zu betrachten. Der Operator erstellt mit Hilfe der Anwendung zur Kommandierung (*Command Ground Application*) ein Kommando, und überreicht dieses der Kommunikationsanwendung (*COM Ground Application*) zur Übertragung an den Satelliten. Der zuletzt genannte Vorgang erfordert mehrere Teilschritte, die mit Hilfe der in Abbildung 7-52 dargestellten

Komponenten durchgeführt werden. Zuerst wird das Kommando durch die Protokollmaschine (*Frame Protocol Maschine*) gemäß einem zuvor in ihr eingestellten Protokoll zu einem Datenpaket (mit Header und Footer) umschlossen und ggfs. segmentiert. In der Praxis haben sich hierbei einige Standards etabliert (z.B. CCSDS). Der letzte Schritt der Protokollmaschine besteht darin, die verpackten Daten für die Weiterleitung an das Modem (*Terminal Node Controller, TNC*) aufzubereiten. Hierbei erfolgt in der Regel eine letzte Segmentierung der Daten-Pakete in Daten-Frames (OSI-Level 2). Das Modem wird dann aus den Frames die Datensignale erzeugen und auf einen Signalträger aufmodulieren. Zum Schluss erfolgt die Übergabe des Signals an einen Übertrager (*Transmitter*). Dieser sendet letztendlich das Signal mit Hilfe seiner internen Verstärker und seiner angeschlossenen Antenne über das Medium (Erdatmosphäre und Vakuum) an den Bestimmungsort, d.h. den Satelliten. Dort werden die empfangenen Datensignale mit Hilfe der dort ansässigen Kommunikationsanwendung (*COM Application*) wieder zurück in ein komplettes Kommando zusammengesetzt. Dabei laufen die bisher auf der Senderseite genannten Prozessschritte auf der Empfängerseite in umgekehrter Reihenfolge und in invertierter Weise ab. Da die Kommunikation innerhalb des Satelliten auf der RODOS-Middleware aufsetzt, wird das empfangene Kommando nach seiner Wiederherstellung anschließend in eine RODOS-Botschaft eingebettet und an die Anwendung zur Kommandoverwaltung weitergereicht. Diese wird die Botschaft mit dem darin enthaltenen Kommando zur gegebenen Zeit (immediate oder time-tagged) an die APS- Anwendung weiterleiten.

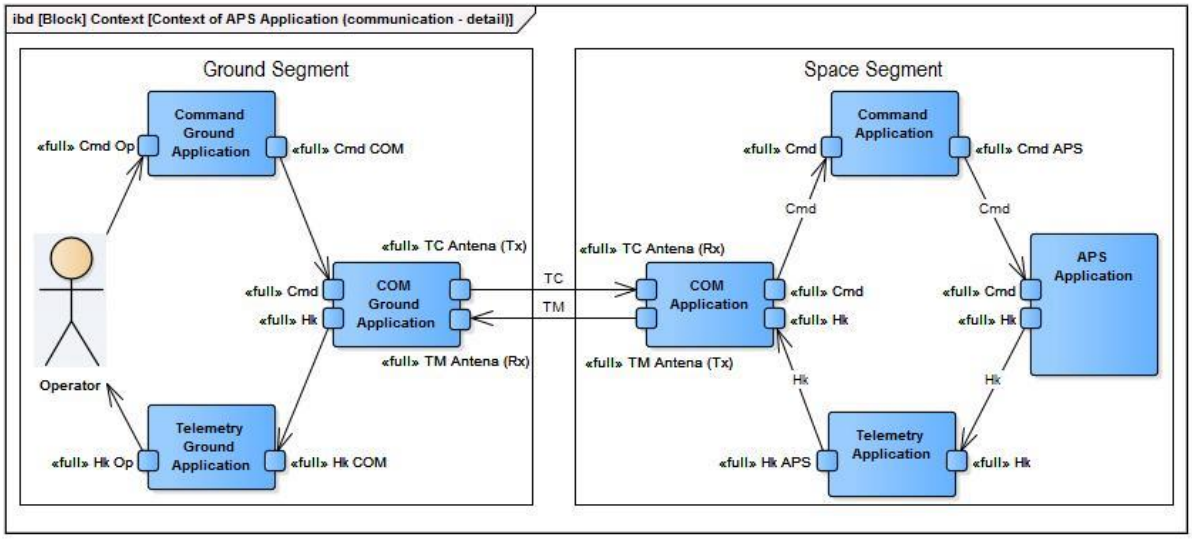


Abbildung 7-51: Kommunikation zwischen Operator und APS (detaillierte Darstellung)

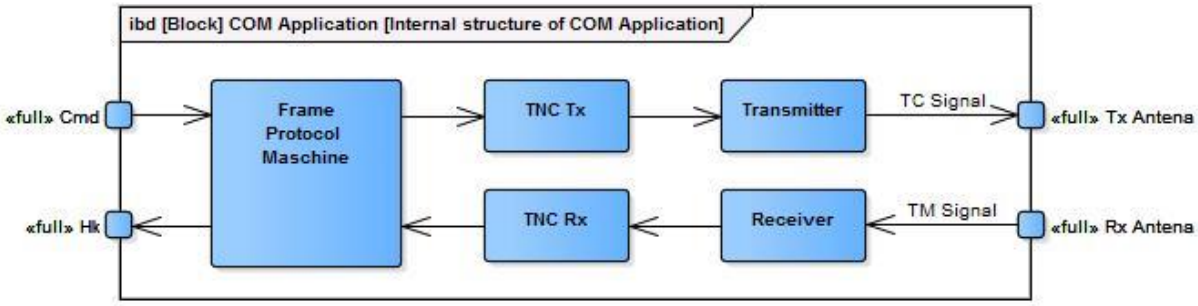


Abbildung 7-52: Interner Aufbau einer COM-Anwendung

Im Hinblick auf die Telemetrie (siehe Abschnitt 2.2.4.3) wird der Operator insbesondere an Real-Time-Telemetrie und History-Telemetrie zum Zwecke der Überwachung und an Extended-Telemetrie zur Konfiguration des autonomen Verhaltens interessiert sein. Sollte sich das Protokoll der Kommunikation am CCSDS – Standard orientieren, so ist es empfehlenswert, sich der darin vorgesehenen „virtuellen Kanäle“ zu bedienen und den Datenstrom entsprechend aufzuteilen.

7.3 ASAP

Dieser Abschnitt beschäftigt sich mit ASAP (siehe Abbildung 7-53), einer Komponente für Satelliten zur optischen Erfassung von Phänomenen und zur Umsetzung autonomer Funktionalitäten. Es handelt sich hierbei um ein eingebettetes System, bestehend aus Hardware und Software, und wurde nach der im Kapitel 5 beschriebenen Methode entwickelt.

Abbildung 7-53: ASAP (Urheber: Oleskii Balagurin)

ASAP besteht aus drei verschiedenen Teilsystemen:

- Hardwareplattform
- autonomes Planungssystem (APS)
- autonomes Sensorsystem (ASS)

Diese Teilsysteme werden in den folgenden Unterabschnitten detaillierter betrachtet. Im Abschnitt 7.2 wurde die Funktionsweise des APSs bereits beschrieben. Auf sie wird in diesem Abschnitt daher nicht näher eingegangen. Stattdessen wird die Anwendung (d.h. die Konfiguration anhand der Referenzszenarien) des APSs im Vordergrund stehen.

Zum Zeitpunkt der Fertigstellung der vorliegenden Dissertation befindet sich ASAP noch in der Entwicklung. Momentan laufen Arbeiten, um eine reduzierte Variante von ASAP in das Satellitenvorhaben SONATE zu integrieren.

Die nachfolgenden Unterabschnitte beschreiben die einzelnen Zwischenschritte des Entwicklungsprozesses.

7.3.1 Zielstellung

Die Hauptaufgabe von ASAP besteht in der Detektion der zuvor im Abschnitt 7.1 genannten Referenzobjekte und dem Erzeugen und Umsetzen autonomer Entscheidungen im Verbund mit den anderen Subsystemen des Satelliten. Richtungsweisend gelten die in Abschnitt 7.1 dargestellten Referenzszenarien, die es zu realisieren gilt. Insbesondere sollen optische Aufnahmen der detektierten Ereignisse, Phänomene und Ziele aufgezeichnet und zur Bodenstation gesendet werden.

7.3.2 Stakeholder

Die nachfolgende Abbildung 7-54 stellt eine Übersicht über alle Stakeholder, die eine relevante Beziehung zu ASAP haben, dar.

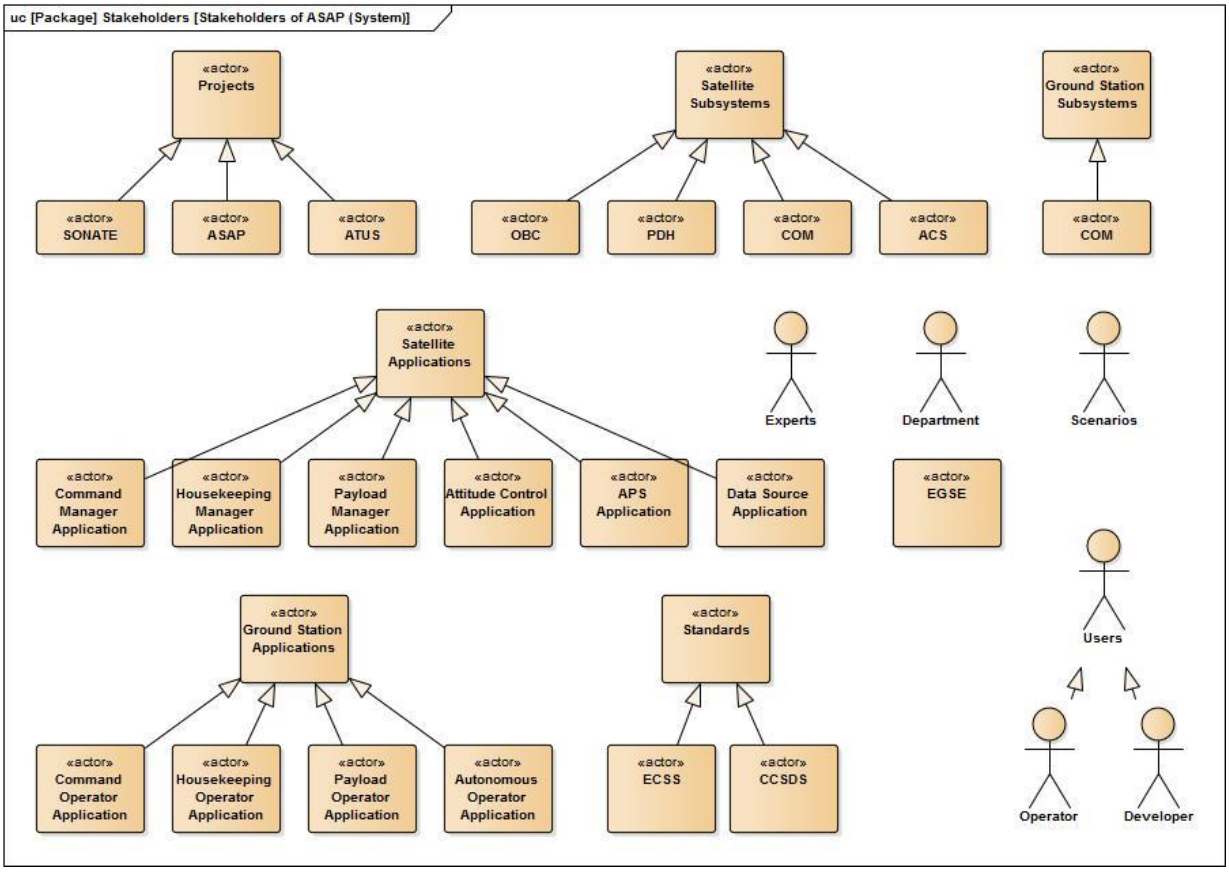


Abbildung 7-54: Stakeholders von ASAP

Ein Teil der Stakeholder wurde schon im Abschnitt 7.2.2 bei der Beschreibung des autonomen Planungssystems (APS) eingeführt. Aus diesem Grund sollen an dieser Stelle nur noch die unbekannten Stakeholder vorgestellt werden.

Wie bereits in der Zielstellung erwähnt wurde, arbeitet ASAP im Verbund mit anderen Subsystemen des Satelliten:

<u>Subsystemen des Satelliten</u> (Satellite Subsystems)

- Bordcomputer des Satelliten (Onboard Computer, OBC)
  Der Bordcomputer ist die Hardwareplattform, auf der die die Anwendungen zur Kommando- und Housekeepingverwaltung des Satelliten laufen. ASAP muss mit dem OBC interagieren, um autonom in die Steuerung des Satelliten eingreifen zu können.
- Nutzlaststeuerung des Satelliten (Payload Data Handling, PDH)
  Die PDH ist die Hardwareplattform, die zur Ansteuerung der Instrumente und zur Verwaltung der Nutzlastdaten eines Satelliten benötigt wird. ASAP, bzw. das ASS, ist als eine Nutzlast anzusehen und muss daher mit dem PDH kommunizieren.
- Kommunikationssteuerung (Communication, COM)
  Das COM System regelt den Datenaustausch (Telekommandos und Telemetrie) zwischen Satellit und Bodenstation.
- Lageregelungssteuerung (Attitude Control System, ACS)
  Das ACS regelt die räumliche Ausrichtung des Satelliten im Weltall. Einige Referenzszenarien beinhalten die Verwendung des ACS.

Trotz der autonomen Ausrichtung einer Satellitenmission ist beim Missionsbetrieb die Bodenstation in die Missionsabläufe eingebunden. Zu diesem Zweck müssen folgende Anwendungen als Stakeholder berücksichtigt werden:

Anwendungen der Bodenstation (Ground Station Applications)

- Kommandoverwaltung für den Satelliten (Command Operator Application)
  Der Operator wird ASAP über eine entsprechende Schnittstelle steuern.
- Housekeepingverwaltung (Housekeeping Operator Application)
  Der Operator von ASAP wird über eine entsprechende Schnittstelle die Housekeeping-Daten abrufen.
- Nutzlastdatenverwaltung (Payload Operator Application)
  Der Operator wird von ASAP über eine entsprechende Schnittstelle die Nutzlastdaten abrufen.
- Autonomieverwaltung (Autonomous Operator Application)
  Der Operator wird von ASAP über eine entsprechende Schnittstelle Daten über den autonomen Betrieb einsehen.

In der Raumfahrt haben sich einige Standards durchgesetzt, die es bei ASAP zu berücksichtigen gilt:

Standards

- European Cooperation on Space Standardization (ECSS)
  Der ECSS-Standard definiert Richtlinien und Empfehlungen, die ASAP betreffen.
- The Consultative Committee for Space Data Standardization (CCSDS)
  Der CCSDS-Standard definiert Richtlinien und Empfehlungen, die ASAP betreffen.

Sonstige Stakeholder von ASAP:

Sonstiges

- Electronic Ground Support Equipment (EGSE)
  Zu Test- und Betriebszwecken muss ASAP an ein EGSE angeschlossen werden.

7.3.3 Anforderungen

Eine Analyse der Beziehungen und Abhängigkeiten zwischen dem zu entwickelnden ASAP System und seinen Stakeholdern hat einen Satz von Nutzeranforderungen und Systemanforderungen, untergliedert in funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen, aufgezeigt. Sie werden in den nachfolgenden Tabellen aufgeführt. Die Anforderungen lassen sich zu inhaltlichen Gruppen, die jeweils einem Stakeholder zugeordnet sind, zusammenfassen. Elementare Anforderungen werden nach Möglichkeit begründet.

Die nachfolgende Tabelle 7-11 listet die Nutzeranforderungen an das ASAP System:

ID	Name und Begründung
1	Nutzeranforderungen
1.1	Anforderungen seitens der Referenzszenarien
1.1.1	Optische Erfassung ASAP muss in der Lage sein, die im Abschnitt 7.1 beschriebenen Referenzobjekte optisch zu erfassen.
1.1.2	Autonome Handlung ASAP muss zur Erreichung der in den Referenzszenarien beschriebenen Ziele autonom Entscheidungen treffen und umsetzen.
1.1.3	Interaktion mit dem Umfeld (Kontext) Zur Umsetzung der autonomen Handlung muss ASAP mit dem Umfeld (z.B. Boardcomputer und Nutzlastdatencomputer) des Satelliten interagieren.

1.2	Anforderungen seitens der Satelliten-Subsysteme
1.2.1	Mechanische Integration ASAP muss sich mechanisch (z.B. an seinem Gehäuse) in die tragende Struktur des Satelliten integrieren lassen.
1.2.2	Elektrische Integration ASAP muss sich elektrisch (z.B. Versorgung mit Energie und Signalen) an die Bordelektronik des Satelliten anschließen lassen.
1.2.3	Optische Integration ASAP muss sich optisch (z.B. Ausrichtung und Kalibrierung der Kamera) in den Satelliten integrieren lassen.
1.3	Anforderungen seitens der Satelliten-Anwendungen
1.3.1	Lageerfassung ASAP muss seine eigene Housekeeping-Daten (z.B. innere Betriebszustände) an die Anwendung zur Lageerfassung des Satelliten senden.
1.3.2	Kommandierung ASAP muss an ihn gerichtete Kommandos empfangen, interpretieren und ausführen können.
1.3.3	Nutzlastdatenverwaltung ASAP muss seine gewonnenen Nutzlast-Daten (z.B. Bilder) an die Anwendung zur Nutzlastdatenverwaltung des Satelliten senden.
1.4	Anforderungen seitens des Operators
1.4.1	Betriebsmodi ASAP muss mindestens folgende Betriebsmodi unterstützen: Init : Eigenständiges Booten und Konfigurieren von ASAP Passiv : Operativer Betrieb ohne autonome Interaktionen (z.B. kann die Kamera manuell bedient werden) Halb-Autonom: Operativer Betrieb mit autonomen Funktionen, aber ohne Auswirkungen (d.h. Kommandogenerierung) auf das Verhalten des Satelliten Voll-Autonom : Operativer Betrieb mit allen autonomen Funktionen und voller Rückkopplung auf das Verhalten des Satelliten
1.4.2	Housekeeping Daten (Zustand) ASAP muss seine Housekeeping-Daten sowohl eigenständig (Standard Housekeeping Online/Offline) als auch auf Anforderung durch den Operator (Extended Housekeeping) generieren können.
1.4.3	Housekeeping Daten (Autonome Planung) ASAP muss insbesondere „Snapshots“ über den Zustand der autonomen Planung (z.B. geplante Aktivitäten, registrierte Ereignisse, Kommandoliste) speichern, mit Zeitstempeln versehen, und als Extended Housekeeping zum Boden übertragen können.
1.4.4	Housekeeping Daten (Autonome Bildverarbeitung) ASAP muss bei einer Detektion (Ereignis oder Phänomen) beschreibende Informationen (Meta Daten) aus den Bilddaten extrahieren. Die wichtigsten Informationen (Anzahl, Typ, Bewegung) müssen per Online-Telemetrie übertragen werden. Erweiterte Informationen sollen per Extended-Telemetrie abrufbar sein.
1.5	Anforderungen seitens des Entwicklers (Autors)
1.5.1	Konfiguration Das Verhalten von ASAP (insb. das autonome Verhalten) muss einstellbar sein. Ein Software-Update sollte zu diesem Zweck nicht nötig sein.
1.5.2	Flexibilität Die Konfiguration sollte vollständig per Kommandierung vorgenommen werden können.
1.6	Anforderungen seitens des Lehrstuhls
1.6.1	Software Update Ein Software-Update von ASAP im Orbit soll möglich sein.
1.6.2	Meta-Daten Die von ASAP aus den Nutzlastdaten extrahierten Meta-Daten müssen folgende Eigenschaften aufweisen.
1.6.2.1	Extraktion Bei einer Detektion eines Ereignisses/Phänomens sollen der Typ, die Anzahl, und der Bewegungs-Vektor extrahiert werden.

1.6.2.2	Verfügbarkeit Die Meta-Daten sollen innerhalb von 24 Stunden nach der Entdeckung zum Boden gesendet werden.
1.6.3	Nutzlastdaten
1.6.3.1	Nicht-flüchtige Speicherung ASAP soll die gewonnenen Nutzlastdaten (unabhängig von einer externen Nutzlastdatenverwaltung) nicht-flüchtig speichern können.
1.6.3.2	Verfügbarkeit Die Nutzlastdaten sollen innerhalb von 48 Stunden nach der Entdeckung zum Boden gesendet werden.

Tabelle 7-11: Nutzeranforderungen an das ASAP System

Die nachfolgende Tabelle 7-12 listet die funktionalen Anforderungen an das ASAP System:

ID	**Name und Begründung**
2	Systemanforderungen (funktionale Anforderungen)
2.1	Anforderungen seitens der Referenzszenarien
2.1.1	Autonome Erfassung ASAP muss über eine autonome Erfassung verfügen, um die Referenzszenarien realisieren zu können.
2.1.1.1	Ziele der autonomen Erfassung ASAP muss im Rahmen der autonomen Erfassung folgende Ziele optisch selbstständig erkennen und auswerten können.
2.1.1.1.1	Muster ASAP muss bestimmte Muster optisch selbstständig erkennen und auswerten können.
2.1.1.1.2	Formen ASAP muss bestimmte Formen (z.B. Blitz vs. Sprite vs. BlueJet) optisch selbstständig erkennen und auswerten können.
2.1.1.1.3	Objekte ASAP muss bestimmte Objekte (z.B. Meteore) optisch selbstständig erkennen und auswerten können.
2.1.1.1.4	Bewegungen ASAP muss bestimmte Bewegungen (z.B. Trajektorien) optisch selbstständig erkennen und auswerten können.
2.1.1.1.5	Ereignisse ASAP muss bestimmte Ereignisse/Phänomene (z.B. Blitze) optisch selbstständig erkennen und auswerten können.
2.1.1.2	Ausgänge der autonomen Erfassung ASAP muss im Rahmen der autonomen Erfassung folgende Informationen ausgeben.
2.1.1.2.1	Meta Daten ASAP muss aus den Eingangsdaten Meta-Daten ableiten und zur autonomen Planung weiterreichen.
2.1.1.2.2	Nutzlast Daten ASAP muss aus den Eingangsdaten Nutzlast-Daten ableiten, zwischenspeichern und bei Aufforderung an das PDH- oder COM-Subsystem des Satelliten weiterreichen.
2.1.2	Autonome Planung ASAP muss über eine autonome Planung verfügen, um die Referenzszenarien realisieren zu können.
2.1.2.1	Eingänge der autonomen Planung ASAP muss im Rahmen der autonomen Planung folgende eingehende Daten verarbeiten.
2.1.2.1.1	Meta Daten ASAP muss Meta-Daten einlesen und verarbeiten können. Insbesondere sollen dadurch autonome Entscheidungsprozesse angestoßen werden.
2.1.2.2	Ausgänge der autonomen Planung ASAP muss im Rahmen der autonomen Planung folgende Daten ausgeben.
2.1.2.2.1	Kommandos ASAP muss autonom Kommandos zur Steuerung des Satelliten erstellen.

2.1.2.3	Verhalten der autonomen Planung ASAP muss bei der autonomen Planung folgende Verhaltensweisen aufzeigen.
2.1.2.3.1	Analysieren ASAP muss eingehende Daten (z.B. Meta-Daten) analysieren.
2.1.2.3.2	Entscheiden ASAP muss auf der Grundlage der Analyse Entscheidungen fällen, die das Verhalten des Satelliten betreffen.
2.1.2.3.3	Ablaufplanen ASAP muss Abläufe planen, da die Ressourcen des Satelliten (z.B. Speicher, Energie) beschränkt sind.
2.1.2.3.4	Ausführen ASAP muss die getroffenen Entscheidungen und geplanten Abläufe umsetzen, indem es Kommandos für den Satelliten generiert und an diesen überträgt.
2.1.2.4	Dienste ASAP muss im Rahmen der autonomen Planung auf folgende Dienste zugreifen.
2.1.2.4.1	Orbit Dienst ASAP muss auf einen Dienst zugreifen, der Berechnungen zu Orbit-Informationen (z.B. Position) liefert.
2.2	Anforderungen seitens der Satelliten-Subsysteme
2.2.1	OBC
2.2.1.1	Schnittstelle ASAP muss eine Schnittstelle zum OBC aufweisen.
2.2.2	PDH
2.2.2.1	Schnittstelle ASAP muss eine Schnittstelle zum PDH aufweisen.
2.3	Anforderungen seitens der Satelliten-Anwendungen
2.3.1	Kommandoverwaltung
2.3.1.1	Interaktionen ASAP muss mit der Kommandoverwaltung interagieren.
2.3.1.1.1	Synchronisieren der Kommandos ASAP muss seine im Rahmen der autonomen Planung generierten Kommandos an die Kommandoverwaltung zur Ausführung übergeben. Bei der Übertragung muss ein geordnetes Synchronisierungsverfahren eingesetzt werden.
2.3.1.1.2	Auswerten der Rückmeldungen ASAP muss im Rahmen der autonomen Planung Rückmeldungen von der Kommandoverwaltung auswerten.
2.3.1.2	Interpretieren der Kommandos ASAP muss die an ihn adressierten und von der Kommandoverwaltung übermittelten Kommandos interpretieren.
2.3.2	Housekeepingverwaltung
2.3.2.1	Synchronisieren der Housekeeping ASAP muss eine eigene Kopie der Housekeeping-Daten des Satelliten verwalten und synchronisieren. Diese Daten können je nach Konfiguration in die autonomen Planungsprozesse miteinbezogen werden.
2.3.2.2	Erstellen der eigenen Housekeeping ASAP muss seine eigenen Housekeeping-Daten erfassen und zur Housekeeping-Verwaltung übermitteln.
2.3.3	Nutzlastdatenverwaltung
2.3.3.1	Bereitstellen von Nutzlastdaten ASAP muss seine eigenen Nutzlast-Daten erfassen und zur Nutzlastverwaltung übermitteln.
2.4	Anforderungen seitens des Operators
2.4.1	Überwachen ASAP soll es dem Operator ermöglichen, dass dieser seinen Betrieb überwachen kann.
2.4.2	Konfigurieren ASAP soll es dem Operator erlauben, sein autonomes Verhalten je nach

	gewünschtem Szenario einstellen zu können. Dies soll ausschließlich per Kommandierung erfolgen.
2.4.3	Betrieb
2.4.3.1	Betriebsmodi ASAP soll es dem Operator ermöglichen, zwischen einem der drei möglichen Betriebsmodi zu wählen: - Konventioneller Betrieb Keine autonomen Funktionen - Geschlossener autonomer Betrieb Missionsbetrieb mit autonomer Planung durch ASAP mit tatsächlichem Eingriff in die Kommandoverwaltung des Satelliten - Offener autonomer Betrieb Missionsbetrieb mit autonomer Planung durch ASAP, aber ohne tatsächlichen Eingriff in die Kommandoverwaltung des Satelliten.
2.5	Anforderungen seitens des Autors
2.5.1	Features ASAP soll folgende Eigenschaften besitzen.
2.5.1.1	Konfigurierbarkeit ASAP soll es ermöglichen, sich an unterschiedliche Szenarien per Kommandierung anzupassen. Dadurch wird das autonome Verhalten geändert.
2.5.1.2	Nachverfolgung der Abläufe ASAP soll seine internen Abläufe protokollieren, damit die autonomen Entscheidungsprozesse nachvollzogen werden können.

Tabelle 7-12: Systemanforderungen (funktionale Anforderungen) an das ASAP System

Die nachfolgende Tabelle 7-13 listet die Qualitätsanforderungen an das APS.

ID	Name und Begründung
3	Systemanforderungen (Qualitätsanforderungen)
3.1	Anforderungen seitens der Referenzszenarien
3.1.1	Schnelligkeit der Planänderungen und Planumsetzungen ASAP soll innerhalb 1 Sekunde auf nicht-deterministische kurzfristige Ereignisse (z.B. Meteor) reagieren können.

Tabelle 7-13: Systemanforderungen (Qualitätsanforderungen) an das ASAP System

Die nachfolgende Tabelle 7-14 listet die Randbedingungen an das APS

ID	Name und Begründung
4	Systemanforderungen (Randbedingungen)
4.4	Anforderungen seitens der Vorhaben/Projekte
4.4.1	RODOS ASAP muss die RODOS-Plattform implementieren und seinen Software-anwendungen entsprechende Schnittstellen anbieten, da davon auszugehen ist, dass es in einem Netzwerk mit RODOS-Knoten (z.B. OBC) eingebettet wird.
4.4.2	ASAP Anforderungen seitens des Vorhabens ASAP.
4.4.2.1	Masse ASAP soll maximal eine Masse von 6 Kg aufweisen.
4.4.2.2	Volumen ASAP soll die räumlichen Abmessungen 25cm x 25cm x 25cm nicht überschreiten.
4.4.2.2	Elektrische Leistung ASAP soll im Durchschnitt nicht mehr als 10W elektrische Leistung benötigen.

Tabelle 7-14: Systemanforderungen (Randbedingungen) an das ASAP System

7.3.4 Systemkontext

In den vorherigen Abschnitten wurden die Stakeholder identifiziert und die Anforderungen an das ASAP System analysiert. Auf der Grundlage dieser Vorarbeiten kann nun im nächsten Schritt der Kontext des zu entwickelnden Systems abgegrenzt werden (siehe Abbildung 7-55). Damit wird zum einen eine erste strukturierte Ordnung zwischen den beteiligten Systemen eingeführt, zum anderen wird jetzt klarer ersichtlich, welche Teilnehmer direkt miteinander interagieren und somit eine gemeinsame Schnittstelle aufweisen müssen.

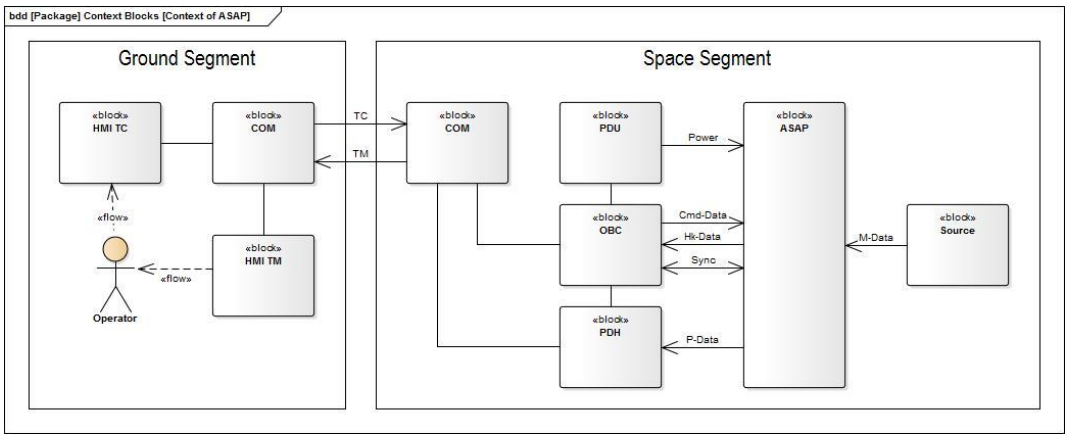


Abbildung 7-55: Systemkontext von ASAP

Generell wurde der Kontext in die zwei Systemgruppen *Ground Segment* und *Space Segment* aufgeteilt. Der Großteil des autonomen Verhaltens wird durch die Interaktionen zwischen ASAP und dem Onboard-Computer (OBC) des Satelliten realisiert. Die Anwendungen des OBC sind unter anderem für die Verwaltung und die Weiterleitung der Kommandos an die einzelnen Komponenten des Satelliten, darunter auch das ASAP System, sowie für die Erfassung der Housekeeping-Daten von den jeweiligen Komponenten, zuständig. Aus den im obigen Abschnitt genannten Anforderungen ist ersichtlich, dass der OBC und das ASAP System eine Synchronisierung eingehen müssen, damit die autonom getroffenen Entscheidungen umgesetzt werden können. Insbesondere müssen dabei Kommandos und Housekeeping-Daten abgeglichen werden. Ein weiterer Aspekt der autonomen Planung betrifft das Erfassen von Meta-Daten. ASAP verfügt durch sein autonomes Sensorsystem über eine eigene Quelle, die Meta-Daten produziert. Es muss jedoch durchaus in Betracht gezogen werden, dass Meta-Daten von externen Quellen eingelesen werden sollen. Dementsprechend ist eine Schnittstelle vorzusehen. Des Weiteren nimmt ASAP auch die Rolle einer Nutzlast (siehe Anforderungen bzgl. autonomen Sensors) im Satelliten ein und generiert entsprechend Nutzlastdaten, die zur Verwaltung an das Payload-Data-Handling System (PDH) übertragen werden müssen.

Damit wären die Interaktionen mit den von ASAP unmittelbar betroffenen Systemen erläutert. Mittelbar betroffene Teilnehmer befinden sich vor allem auf dem Bodensegment. Der Operator als Anwender bedient sich zur Interaktion mit dem Satelliten in der Regel zweier Systeme, einer zur Kommandierung des Satelliten (HMI TC) und einer zur Erfassung der Housekeeping-Daten des Satelliten (HMI TM). Diese Systeme arbeiten mit Datenpaketen, in denen Telekommandos (TC) und Telemetrie (TM) kodiert bzw. dekodiert werden. Der Transfer dieser Datenpakete zwischen den beiden Raumfahrtsegmenten per Funkstrecke wird mit Hilfe spezieller Kommunikationssysteme (COM) koordiniert. Diese implementieren speziell dafür vorgesehene Übertragungsprotokolle, und übersetzen TCs in Kommandos (Cmd) und TMs in Housekeeping-Daten (Hk-Data) und Payload-Daten (P-Data).

7.3.5 Systemsynthese

Der nächste Schritt im Entwurfsprozess für das zu entwickelnde System ist die Synthese. Dabei soll, ausgehend vom Systemkontext und den an ihn gestellten Anforderungen, sein funktionales Verhalten (siehe Abbildung 7-56) realisiert werden.

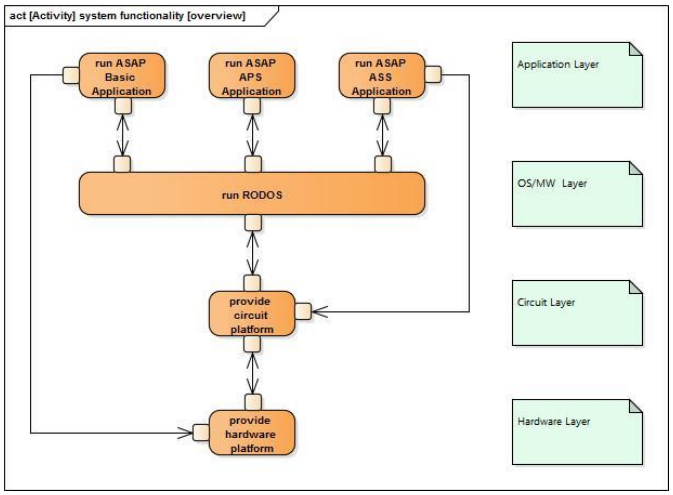


Abbildung 7-56: wesentliche funktionale Aspekte des ASAP Systems

Die Funktionalität von ASAP wurde auf vier Schichten verteilt. Die oberste Schicht ist die Anwendungsschicht. In ihr befinden sich drei Anwendungen. Das Hauptziel von ASAP besteht darin, mit den zwei Hauptanwendungen APS (Autonomes Planungssystem) und ASS (Autonomes Sensor System) auf die in den Referenzszenarien dargestellte Weise wissenschaftliche Daten für die Forschung zu gewinnen. Sie realisieren die im Abschnitt 7.3.3 gestellten Anforderungen der autonomen Planung und der autonomen Erfassung. Das APS wurde im Abschnitt 7.2 vorgestellt, und dient dazu, den Satelliten autonom zu steuern. Es zeichnet sich durch seine Konfigurierbarkeit aus, sodass es per Kommandierung an beliebige Szenarien angepasst werden kann. Das ASS ist eine Anwendung, deren primäre Aufgabe es ist, Meta-Daten aus optischen Beobachtungen (die über die ASAP Hardware bereitgestellt werden) zu generieren und an das APS zu schicken. Bei den Meta-Daten handelt es sich um Informationen, die besondere Tatsachen beschreiben (im Falle der Referenzszenarien sind es z.B. Blitze, Meteore, usw.). Das APS reagiert auf diese Meta-Daten, indem es sie autonom auswertet und ggf. als Folge das Verhalten des Satelliten mittels Manipulation der Kommandolisten ändert. Bei der dritten und letzten als *Basic* bezeichneten Anwendung handelt es sich um eine Unterstützungssoftware für die Hardware. Sie dient dazu, rudimentäre Aufgaben wie zum Beispiel die Messwerterfassung der auf der Hardware verbauten Sensoren zu übernehmen. Alle drei vorgestellten Anwendungen haben die Eigenschaft, dass es sich bei ihnen um RODOS Anwendung handelt, die auf dem RODOS Betriebssystem laufen, und untereinander über die RODOS Middleware kommunizieren. RODOS befindet sich in der Hierarchie eine Stufe unter der Anwendungsschicht. Bei den beiden bisher vorgestellten Schichten handelt es sich ausschließlich um Software, die jedoch eine Plattform in Form von elektrischen Schaltkreisen (z.B. einen Prozessor, Controller) zur Ausführung benötigt. Diese ist in der nächst tiefer gelegenen Schicht angesiedelt. Auf unterster Ebene existiert die Hardware, welche die tatsächlichen Ressourcen (z.B. IC-Chips, Steckverbinder) bereitstellt.

An dieser Stelle soll darauf hingewiesen werden, das RODOS eine Abstraktion der Software von der Hardware ermöglicht. Aus diesem Grund ist es möglich, die APS Anwendung auch auf einer anderen Hardware (z.B. den OBC) zu portieren. Bei den anderen beiden Anwendungen (*ASAP Basic Application* und *ASAP ASS Application*) ist dies nicht möglich. Sie sind an die ASAP Hardware gebunden, da sie aus Performancegründen zum Teil die RODOS Abstraktion umgehen und direkt auf die Hardware zugreifen.

7.3.5.1 Hardware

Die Hardware[3] von ASAP dient als Plattform für die in der Systemsynthese eingeführten Softwareanwendungen *ASAP Basic, ASAP APS und ASAP ASS* und muss dementsprechend die benötigten Hardware-Ressourcen bereitstellen. Bedingt durch die gestellten Anforderungen ergibt sich der in der folgenden Abbildung 7-57 dargestellte Entwurf. Maßgebenden Einfluss hatten die Anforderungen zur Detektion und Aufzeichnung der Referenzobjekte.

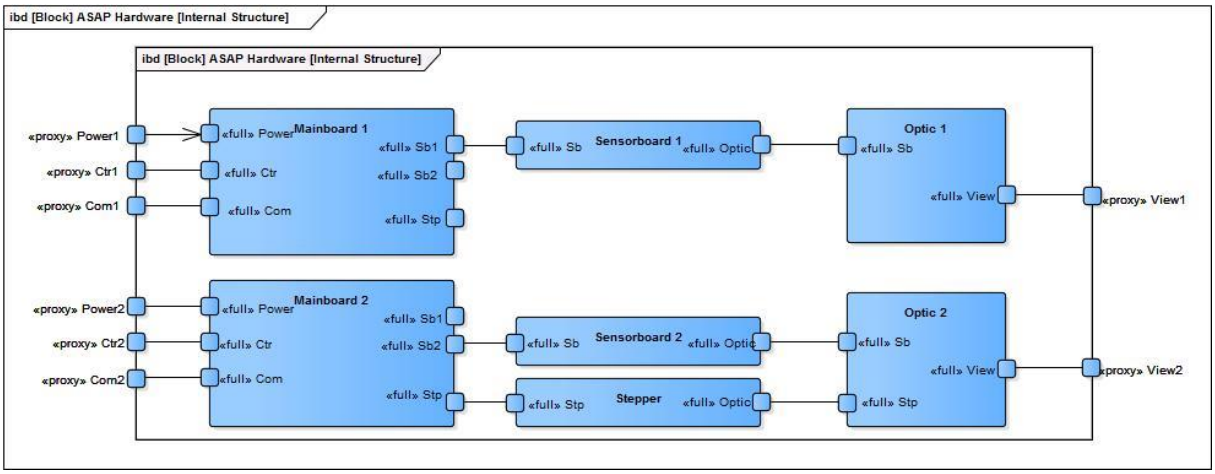


Abbildung 7-57: Interne Struktur von ASAP Hardware

ASAP ist ein duales System. Ein Teilsystem besteht aus einer speziell abgestimmten Optik (Optic) mitsamt zugehörigem Bildsensor, der auf eine dedizierte Platine (Sensorboard) montiert wird und einer daran anschließenden elektronischen Hauptplatine (Mainboard) mit Logik- und Speicherressourcen.

Die Eigenschaften beider optischen Systeme unterscheiden sich stark voneinander, da diese jeweils für unterschiedliche Aufgaben konzipiert wurden. Das erste optische Teilsystem besitzt ein statisches Sichtfeld (FOV, Field Of View) von 38,70° x 38,70° und einen schnellen, leistungsfähigen Bildsensor mit einer Bildwiederholungsrate von mindestens 100 Bildern pro Sekunde und einer Auflösung von 4 MPixel. Damit ist das Teilsystem besonders zum Auffinden von Referenzobjekten im Sichtfeld geeignet. Das zweite optische System ist mit einem schmaleren Sichtfeld von 5,28° x 3,96° und einem langsameren Bildsensor mit einer Bildwiederholungsrate von mindestens 30 Bildern pro Sekunde ausgestattet. Die Besonderheit ist jedoch die Auflösung von 20 MPixel und die Ausstattung mit einer Fokusiereinrichtung, die durch einen Schrittmotor (Stepper) eingestellt wird. Anders als das erste optische System, dessen Aufgabe hauptsächlich im Auffinden der Referenzobjekte besteht, dient das zweite der eigentlichen Beobachtung und Aufzeichnung der Referenzobjekte. Mit ihren unterschiedlichen Eigenschaften ergänzen sich beide optischen Teilsysteme gegenseitig.

Es wurden auch andere Konzepte, zum Beispiel mit nur einer einzigen Optik, deren Sichtfeld durch einen Schrittmotor eingestellt werden kann, ausgearbeitet. Diese haben sich jedoch gegen das vorliegende System nicht durchgesetzt.

Die nachfolgende Abbildung 7-58 zeigt die Hardware von ASAP

[3] Entwicklung, Herstellung und Integration der Hardware fielen nicht ausschließlich in den Verantwortungsbereich des Autors. Insbesondere waren sein Kollege Herr Balagurin (M. Eng.) sowie diverse Unternehmen am Prozess beteiligt. Der eigene Beitrag des Autors an der Hardware wird daher am Ende des Abschnitts klarer abgegrenzt.

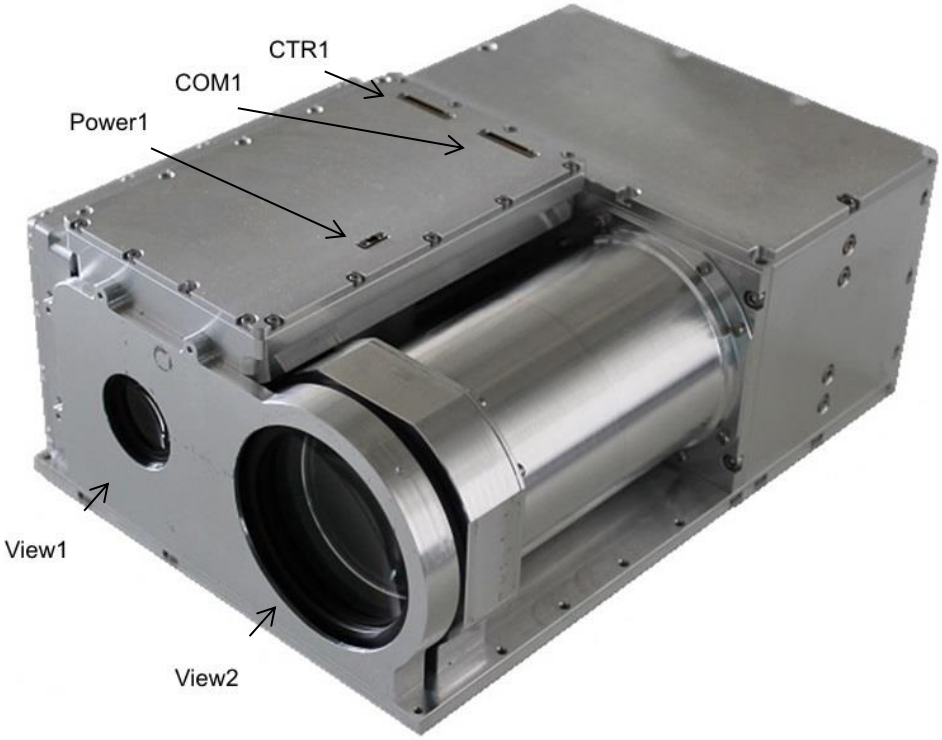


Abbildung 7-58: ASAP Hardware und Schnittstelle (Urheber: Oleskii Balagurin)

Jedes Teilsystem von ASAP ist vom anderen entkoppelt und besitzt daher seine eigene Schnittstelle:

- Power: Energieversorgung 6V – 18V
- CTR: Steuersignale (z.B. Reset) und Low-Speed-Datensignale (z.B. CAN-Bus)
- COM: High-Speed-Datensignale (z.B. Bilddaten)
- View: Optische Fassung

Die einzige gemeinsam genutzte Schnittstelle ist das Gehäuse, in dem beide Teilsysteme montiert sind. Es dient ebenfalls dazu, ASAP mechanisch im Satelliten zu fixieren.

Das optische Gesamtsystem ist konzeptionell in der nachfolgenden Abbildung 7-59 dargestellt.

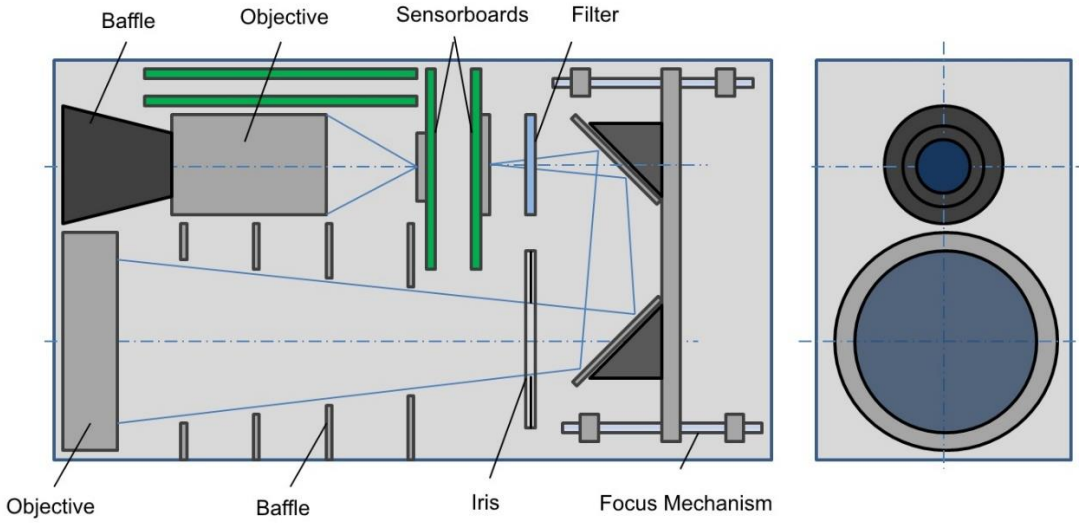


Abbildung 7-59: Konzept der Optik von ASAP (Urheber: Oleskii Balagurin)

Die Anordnung links oben stellt die Optik mit dem weiteren Sichtfeld dar. Ihre Strahlen werden direkt gebündelt und auf den linken Bildsensor gerichtet. Die Konzeption dieses Teilsystems ist relativ einfach. Technisch wird es wesentlich herausfordernder, den Teil der Optik zu realisieren, der dem schmaleren Sichtfeld entspricht. Das Problem hierbei besteht darin, dass der Strahlengang für eine direkte Anordnung der optischen Elemente längs einer einzigen Achse zu lang ist. Die Begrenzungen sind durch die Anforderungen bedingt, insbesondere durch die zur Verfügung stehenden engen räumlichen Verhältnisse. Stattdessen wird der Strahl durch eine Spiegeleinrichtung insgesamt um 180° umgelenkt und trifft im oberen Abschnitt auf den Bildsensor. Zum Zweck der Fokussierung kann der Spiegel zusätzlich längs des Strahlengangs um ein paar Millimeter mit Hilfe eines Schrittmotors bewegt werden.

Jedes optische Teilsystem verfügt über einen auf der Sensorplatine (siehe Abbildung 7-60 und Abbildung 7-61) montierten Bildsensor, der anhand bestimmter Kriterien ausgewählt wurde.

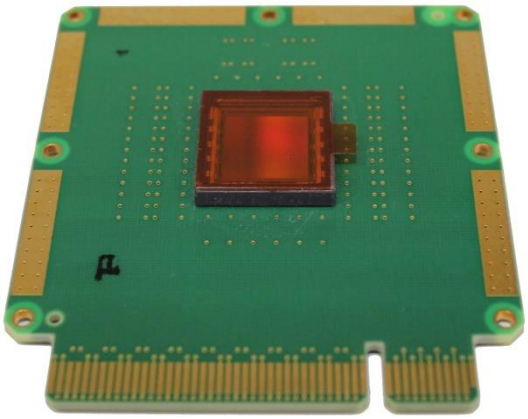


Abbildung 7-60: Sensorplatine 1

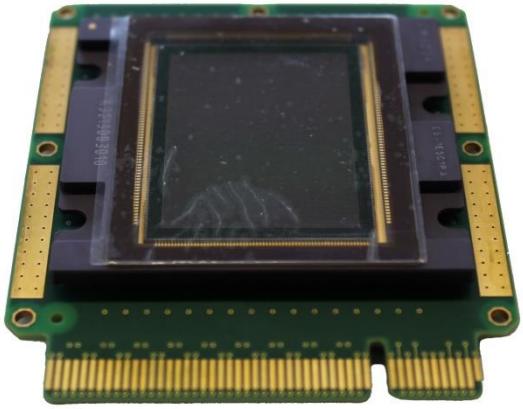


Abbildung 7-61: Sensorplatine 2

Ausgewählt wurden zwei Bildsensoren desselben Herstellers mit zwar unterschiedlichen Eigenschaften, jedoch mit fast identischen elektrischen Schnittstellen:

- Die Parametrierung erfolgt über einen SPI-Bus.
- Die Datenausgabe erfolgt über bis zu 16 LVDS-Datenleitungen.
- Der Sensor benötigt für den normalen Betrieb und die Datenausgabe jeweils einen Taktgeber.

Beide Sensoren produzieren im operativen Betrieb folgenden Datenstrom:

$$Sensor1: 100fps * 4MP * 8Bits = 400MBytes/s$$

$$Sensor2: 30fps * 20MP * 8Bits = 600MBytes/s$$

Um die Power- und Signalintegrität der Schaltung auf den Bildsensorplatinen zu gewährleisten, wurden folgende Maßnahmen ergriffen:

- Auslegung eines Energieversorgungsnetzwerkes für die Bildsensoren. Abblockung durch Kondensatoren.
- Sicherstellung eines differentiellen Wellenwiderstandes von 100Ω für die LVDS-Datenleitungen durch Layoutmaßnahmen (Routing und Stack-Up).
- Differentielle Terminierung der LVDS-Datenleitung verbraucherseitig mit 100Ω.

Die elektrische Verbindung der Bildsensorplatine mit der Hauptplatine wird mit einem PCI-Steckverbinder hergestellt.

Wie bereits oben erwähnt wurde, stellt die Hauptplatine die von den Softwareanwendungen (Basic, APS, ASS) benötigten Ressourcen zur Verfügung. Strukturell lässt sie sich wie in der folgenden Abbildung 7-62 darstellen.

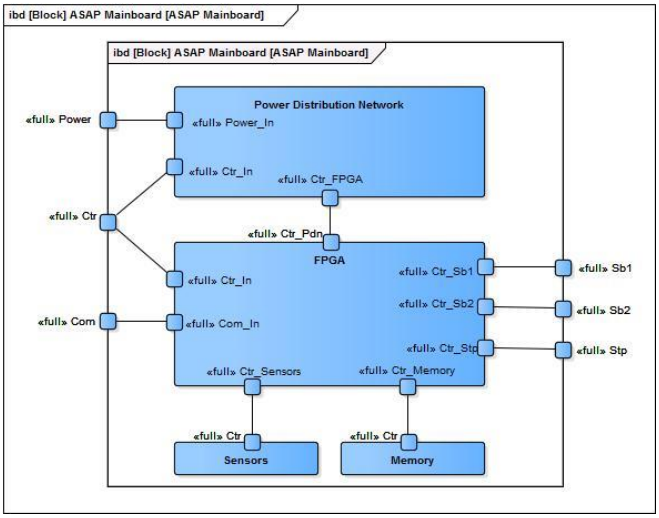


Abbildung 7-62: Interne Struktur der Hauptplatine von ASAP

Jede Hauptplatine verfügt als Schnittstelle nach außen über drei Steckverbinder (Power, Ctr, Com). Diese wurden schon bei der Beschreibung der Schnittstelle des Gesamtsystems (ASAP) genannt. Tatsächlich befinden sich die Anschlüsse aber auf den Hauptplatinen und werden lediglich durch das Gehäuse von ASAP nach außen geführt.

Ebenso wird auch eine Schnittstelle (Sb1, Sb2, Stp) zu den bereits eingeführten optischen Teilsystemen bereitgestellt.

Die bereit gestellten Ressourcen lassen sich in mehrere Gruppen zusammenfassen:

- IC mit programmierbaren Schaltkreisen (FPGA, Field Programmable Gate Array)
  Ein Pool aus Logik-Gattern, die durch Programmierung zu einer Schaltung vernetzt werden können. Damit lassen sich auch kompliziertere Logikschaltungen (z.B. Prozessoren und Controller) realisieren.
- ICs mit Taktgebern (Clock)
- ICs mit Speicher (Memory)
  - 3x 512 MB DDR3-SDRAM
  - 4x 4096 MB Flash
- ICs mit Sensoren (Sensors)
  - 4x Temperatur
  - 9x Spannung/Strom
- IC zur Überwachung (WatchDog)
- Energieversorgungsnetzwerk (Power Distribution Network)
  Schaltkreis bestehend aus mehreren ICs und passiven Bauteilen, die die von den Verbrauchern (z.B. FPGA) benötigten Spannungen und Ströme in gewünschter Qualität (d.h. ohne die Toleranzgrenze durch Schwankungen zu überschreiten) zur Verfügung stellen
  - 0,75V: Vorspannung für DDR3-Speicher
  - 1,20V: Versorgungsspannung für den Kern des FPGA
  - 1,50V: Versorgungsspannung für die DDR3-IOs des DDR3-Speichers/FPGA
  - 1,80V: Versorgungsspannung für den Bildsensor
  - 3,30V: Versorgungsspannung für sonstige IOs (FPGA, Sensoren, …)
  - 5,00V: Versorgungsspannung für den Schrittmotor

Die nachfolgende Abbildung 7-63 zeigt eine Hauptplatine der Hardware von ASAP

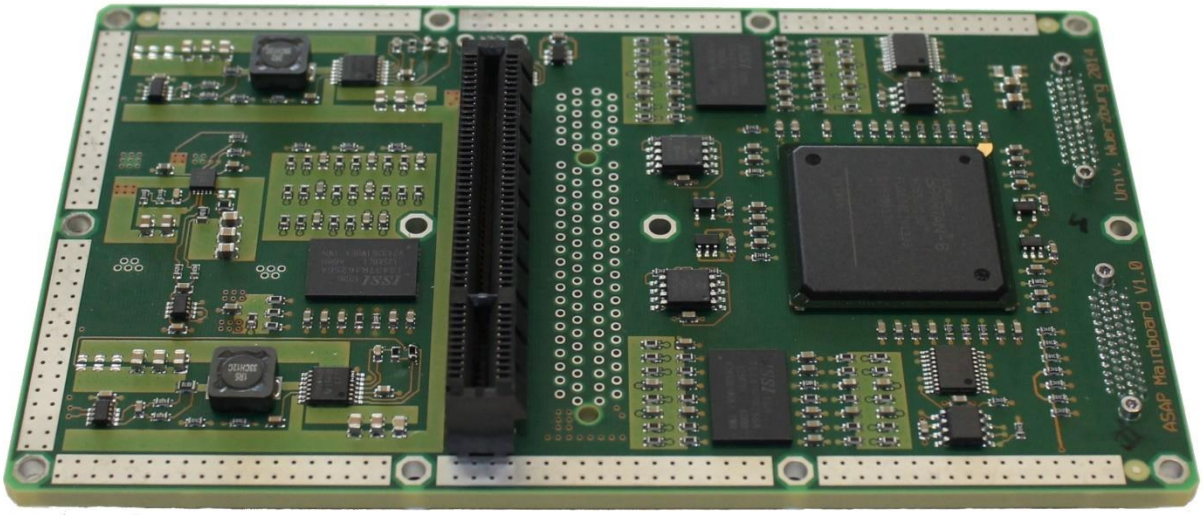


Abbildung 7-63: ASAP Hauptplatine

Um die Power- und Signalintegrität der Schaltung auf der Hauptplatine zu gewährleisten, wurden folgende Maßnahmen ergriffen:

- Auslegung eines Energieversorgungsnetzwerkes für die einzelnen Verbraucher (FPGA, DDR3-RAM, Flash). Abblockung durch Kondensatoren.
- Sicherstellung eines differentiellen Wellenwiderstandes von 100Ω für die LVDS-Datenleitungen (Bildsensor und COM) und DIFF_SSTL-Datenleitungen (DDR3-RAM) durch Layoutmaßnahmen (Routing und Stack-Up).
- Differentielle Terminierung der LVDS- und DIFF_SSTL-Datenleitung verbraucherseitig mit 100Ω.
- Sicherstellung eines seriellen Wellenwiderstandes von 50Ω für die SSTL-Datenleitungen (DDR3-RAM) durch Layoutmaßnahmen (Routing und Stack-Up)
- Serien-Terminierung der SSTL-Datenleitungen quellenseitig mit 50Ω.

Im Anschluss an die Herstellung wurde die Hardware diversen Qualifikationstests unterworfen. Es ist geplant, diese Tests in Zukunft kurz vor der Integration von ASAP in einen Satelliten zu wiederholen, allerdings in abgeschwächter Form (Akzeptanztests). Die Tests haben die Aufgabe, ASAP starken Belastungen mechanischer, thermischer und radioaktiver Art auszusetzen, um seine Eignung für das vorgesehene Umfeld (Weltall) zu beweisen. Die Spezifikation der Testparameter ergibt sich überwiegend aus der vorgesehenen Einsatzdauer und dem –Ort (Orbit). Folgende Testverfahren wurden mit ASAP durchgeführt:

- Mechanische Vibration:
  ASAP wird in Zyklen für die Dauer mehrerer Minuten mit einer mechanischen Schwingung erregt. Zu Beginn eines Zyklus wird die mechanische Eigenfrequenz von ASAP vermessen. Eine Veränderung der Eigenfrequenz weist auf eine mechanische Beschädigung hin. Die Zyklen können variieren. In der Regel werden Zyklen gefahren, bei denen die Frequenz der Schwingung sich mit der Zeit ändert. Mit ASAP werden lineare Frequenzverläufe und zufallsgenerierte Verläufe gefahren. Der Test soll für ASAP einen Start mit der Trägerrakete simulieren.
- Mechanischer Schock:
  ASAP wird mit einem starken mechanischen Impuls erregt. Anders als beim Vibrationstest wird die Energie schlagartig für eine sehr kurze Zeitdauer übertragen. Auch hierbei werden zur Überprüfung Eigenfrequenzen vermessen. Der Test soll für ASAP die Trennung des Satelliten von der Trägerrakete simulieren, die in der Regel durch eine pyrotechnische Sprengung geschieht.

- Thermal-Vakuum:
  ASAP wird in einer isolierten Kammer unter Vakuum verschiedenen Temperaturen ausgesetzt. Die Schwankungen entsprechen dabei den Zyklen der Sonne/Schatten-Phasen des Satelliten im Orbit. Mit Hilfe von Funktionstests kann die Unversehrtheit des Systems überprüft werden. Der Test dient dazu, sowohl die thermische Stabilität von ASAP als auch seine Robustheit gegenüber durch Vakuum verursachten Effekten (z.B. Ausgasung) zu bestätigen.
- Ionisierte Strahlung
  ASAP ist im Orbit ionisierter Strahlung ausgesetzt. Diese führt über die Zeit zu einer Degradation der Elektronik. Mit Hilfe einer Kobalt-60-Strahlenquelle kann dieser Einfluss in einem Test simuliert werden. Dabei erhält ASAP für die Dauer weniger Stunden eine Strahlendosis, die dem gesamten Betriebszeitraum des Systems im Orbit entspricht. ASAP wurde einer Dosis von 10 KRad ausgesetzt. Ein Sicherheitsfaktor ist hierbei miteingeplant worden.
- Hochenergetische Strahlung (mit Teilchen)
  ASAP ist im Orbit hochenergetischer Strahlung ausgesetzt, die im Betrieb zu Logikfehlern in der Elektronik durch den Zusammenstoß eines Teilchen mit einem Halbleiterübergang führen kann. Dabei kann es zu einer flüchtigen, jedoch behebbaren Manipulation der Speicherzelle, einem sogenannten „Upset“ (SEU) oder sogar zu einer permanenten Beschädigung, einem sogenannten „Latch Up“ (SEL) kommen. Mit Hilfe eines Teilchenbeschleunigers können diese Ereignisse in einem Test simuliert werden.

Abgrenzung der Beiträge des Autors:

Wie bereits am Anfang des Abschnitts erwähnt wurde, waren an der Entwicklung, Herstellung und Integration der Hardware mehrere Parteien (Personen und Unternehmen) beteiligt. Die Entwicklung erfolgte ausschließlich intern am Lehrstuhl. Maßgeblich waren der Autor (verantwortlich für die Elektronik) und sein Kollege Herr Balagurin (verantwortlich für die Optik, Mechanik einschließlich Gehäuse) beteiligt. Für die Arbeiten an ASAP wurden folgende Unternehmen beauftragt:

- Gehäusefertigung: Technischer Betrieb (Werkstatt) der Universität Würzburg
- Gehäusebeschichtung: AHC Oberflächentechnik GmbH in Berlin
- Fertigung elektrischer Platinen: Andus GmbH in Berlin
- Bestückung elektrischer Platinen: SMD Lötservice GmbH in Berlin

Die Integration der einzelnen Bauteile von ASAP zum Gesamtsystem erfolgte intern am Lehrstuhl durch Herrn Balagurin. Im Rahmen der Qualitätssicherung (Erarbeitung der Spezifikation durch Herrn Balagurin) wurden im Anschluss noch diverse Tests durchgeführt:

- Bestrahlung TID: Helmholzzentrum in Berlin
- Bestrahlung Hochenergieteilchen: Helmholzzentrum in Berlin
- Mechanische Vibration: Astro- und Feinwerktechnik Adlershof GmbH in Berlin
- Mechanische Schock: Astro- und Feinwerktechnik Adlershof GmbH in Berlin
- Thermal-Vakum: intern am Lehrstuhl (Balugurin und Autor)

7.3.5.2 Schaltkreise

Der Großteil der an ASAP gerichteten Anforderungen bezieht sich auf sein logisches Verhalten. Dieses wird entweder direkt durch elektronische Schaltkreise oder indirekt durch Software (siehe Abschnitt 7.3.5.3) realisiert. Im vorherigen Abschnitt wurden bereits einige Komponenten (z.B. Sensoren), die zum Teil aus elektronischen Schaltkreisen bestehen, vorgestellt. Der Fokus lag hierbei

auf der Hardware, in diesem Fall also auf dem Gehäuse der ICs. Dieser Abschnitt hingegen widmet sich den elektronischen Schaltkreisen[4].

Zu Beginn sollen die wesentlichen Datenströme von ASAP betrachtet werden. Diese stellen den Ausgangspunkt der Schaltkreissynthese dar.

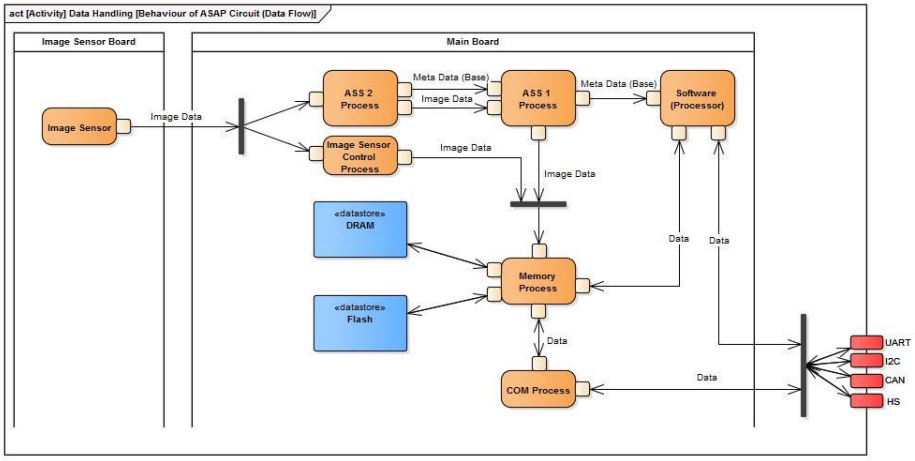


Abbildung 7-64: Verarbeitung der Bilddaten als Hauptfunktionalität von ASAP

Der Fokus liegt dabei auf der Verarbeitung der vom Bildsensor gesendeten Bilddaten (*Image Data*). Diese müssen in einem ersten Schritt (*ASS2 Process*) analysiert werden. Dabei werden Meta-Daten (*Meta Data Base*) extrahiert. Der hierbei gewonnene Informationsgehalt ist als Eingabe für die autonome Planung noch nicht ausreichend. Der Grund liegt darin, dass die Meta-Daten noch keinen räumlichen Bezug auf das Referenzsystem des Satelliten haben. Die gewonnenen Koordinaten beziehen sich lediglich auf das aktuelle Bild und sind dementsprechend zweidimensional. Die weitere Analyse muss daher mit Hilfe von Software auf einem Prozessor (*Software/Processor*) fortgesetzt werden. Ein Zwischenprozess (*ASS1 Process*) dient dabei als Adapter, der die Meta-Daten entgegen nimmt und auf einem der Datenbusse des Prozessors weiterleitet. Gleichzeitig zur Analyse müssen die Bilddaten auch zwischengespeichert werden. Die auftretenden Datenraten (siehe Abschnitt 7.3.5.1) übersteigen die Leistungsfähigkeit des Prozessors, sodass ein anderer, als Schaltkreis realisierter, Prozess (*Memory Prozess*) diese Aufgabe übernehmen muss. Als Speicherort kommen auf Grund des Datenvolumens nur der dynamische RAM (*DRAM*) und der Flashspeicher (*Flash*) in Betracht. Die Speicher sind nicht nur für die Bilddaten vorgesehen, sondern können auch von anderen Softwareanwendungen (z.B. das autonome Planungssystem) genutzt werden. Zur Kommunikation mit anderen Komponenten im Kontext stehen mehrere Datenbusse (z.B. UART, CAN) zur Verfügung. Dabei müssen die entsprechenden Protokolle eingehalten werden. Höhere Übertragungsraten kann der Prozessor ebenfalls nicht handhaben. Zur Entlastung muss ein separater, als Schaltung realisierter, Kommunikationsprozess (*COM Process*) aushelfen.

Als nächstes sollen die Schaltungen gezeigt werden, die das oben gezeigt Verhalten (Datenflüsse) umsetzen. Die Ausgangsbasis bildet die im vorherigen Abschnitt dargestellte Struktur (siehe Abbildung 7-62) der Hardware. Sie kann für die Sicht als Schaltkreis übernommen werden. Dazu werden die einzelnen ICs direkt als Schaltkreise abgebildet. Zu den wichtigsten Komponenten gehören:

- Steckverbinder (Power, CTR, COM)
- 1x FPGA
- 3x 512 MByte DDR3-Speicher

[4] Am Entwurf und der Implementierung der Schaltkreise war ebenfalls Herr Balagurin beteiligt. Der eigene Beitrag des Autors wird daher am Ende des Abschnitts klarer abgegrenzt.

- 4x 4096 MByte Flash-Speicher
- 4x Temperatursensor
- 9x Spannungs-/Stromsensor

Die Steckverbinder (Power, Ctr, Com) der Hauptplatine stellen die Schnittstellen des Schaltkreises stellen dar. Sie bestehen aus folgenden Leitungen:

- Power
  - 6 – 18 V Eingangsspannung
- CTR
  - PowerOn-Signale für die interne Energieversorgung
  - Reset-Signale für die Software-Anwendungen
  - 2x UART-Bus
  - 2x I2C-Bus
  - 2x CAN-Bus
  - Config-JTAG-Bus für die externe Konfiguration der Logik (Debug)
  - Config-SPI-Bus für die externe Konfiguration der Logik
- COM
  - 2x High-Speed-Data zu OBC
  - 2x High-Speed-Data zu PDH

Ein besonderer Baustein ist der FPGA (Field Programmable Gate Array). Im Gegensatz zu den anderen ICs auf der Platine ist seine interne Verschaltung nicht fest vorgegeben, sondern programmierbar. Der Vorgang, bei dem ein Schaltprogramm geladen wird, wird als Konfiguration bezeichnet. Im Grunde besteht dieser Baustein aus elementaren Logik-Gattern, die je nach Programmierung individuell verbunden werden können. Die zur Verfügung stehenden Logik-Ressourcen sind so zahlreich, dass sich damit sogar komplexe Schaltungen (z.B. Prozessoren, Controller) realisieren lassen. Dies hat mehrere Vorteile. Der wichtigste ist jedoch, dass die Schaltung während des Entwicklungsprozesses angepasst werden kann, um zum Beispiel den erhöhten Bedarf an Ressourcen (z.B. eine zusätzliche arithmetische Recheneinheit) seitens der Software zu decken. Zum Einsatz kommt ein FPGA der Baureihe Spartan-6 des Herstellers Xilinx. Seine Konfiguration, d.h. sein Programm, wird aus einem externen Speicher geladen und in einem internen SRAM abgelegt. Dies hat zur Folge, dass sämtliche Konfigurationseinstellungen beim Ausschalten verloren gehen und eine Rekonfiguration beim Einschalten erforderlich wird.

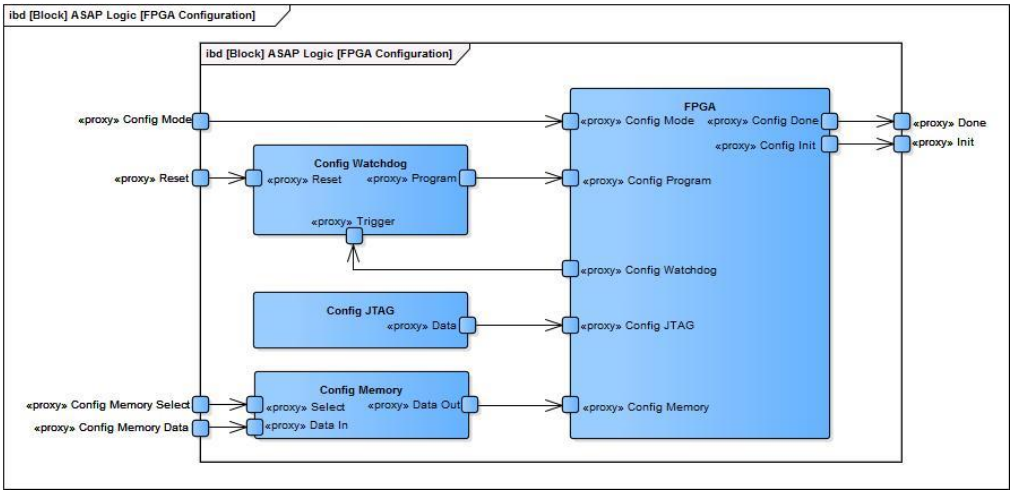


Abbildung 7-65: Konfiguration des FPGAs

Generell existieren mehrere Möglichkeiten, einen FPGA der Baureihe Spartan-6 zu konfigurieren. Die obige Abbildung 7-65 zeigt die in ASAP unterstützten Varianten und weist zusätzlich auf einige Besonderheiten hin.

In ASAP wurden zwei Verfahren zur Konfiguration umgesetzt:

- Aktive Konfiguration über einen externen Speicher
- Passive Konfiguration per JTAG-Programmierer (z.B. Xilinx Platform Cable USB II)

Für den normalen Betrieb ist das erste Konfigurationsverfahren bestimmt. Dieses sieht vor, dass der FPGA sein Konfigurationsprogramm beim Hochfahren eigenständig aus einem externen Speicher bezieht. Dazu müssen von außen (Steckverbinder CTR) einige Einstellungen vorgenommen werden. Zuerst muss der entsprechende Konfigurationsmodus (*Config Mode*) für den FPGA ausgewählt werden. Des Weiteren muss von außen festgelegt werden, aus welchem Speicher (*Config Memory Select*) das Konfigurationsprogramm entnommen werden soll, da aus Redundanzgründen zwei separate Speicher zur Verfügung stehen. Zum Schluss muss noch der Reset (*Reset*) für den FPGA aufgehoben werden. Dies hat zur Folge, dass ein Start-Signal (*Config Program*) an das FPGA gesendet wird, welches die Konfigurationssequenz initiiert. Der Fortschritt der Konfiguration wird vom FPGA mit Hilfe zweier Signale (*Init, Done*) ausgegeben und an den CTR-Steckverbinder weitergeleitet. Das Verfahren wird als aktiv bezeichnet, da der FPGA eigenständig auf den externen Speicher zugreift.

Das zweite Konfigurationsverfahren kann im Grunde jederzeit initiiert werden und benötigt keine Vorkehrungen, da das externe Programmiergerät direkt auf den SRAM-Konfigurationsspeicher im FPGA zugreift. Der Zugriff erfolgt über die JTAG-Schnittstelle. Diese ist über den ASAP-Steckverbinder (CTR) nach außen geführt. Das Verfahren wird als passiv bezeichnet, da die Initiative vom Anwender des Programmiergerätes ausgeht, und dieses eigenständig das Konfigurationsprogramm an den FPGA über JTAG sendet. Dieses Verfahren eignet sich jedoch nur zu Entwicklungs- und Testzwecken, da im normalen Betrieb kein JTAG-Programmierer vorgesehen ist.

Des Weiteren soll noch auf eine zusätzliche Überwachungsschaltung, den WatchDog, hingewiesen werden. Diese greift ebenfalls in die Konfiguration ein. Das Prinzip ist recht einfach, und besteht darin, dass die Überwachungsschaltung vom FPGA in zyklischen Zeitabständen ein Signal (*Trigger*) erwartet. Wenn dieses ausbleibt, geht der WatchDog von einem Fehlverhalten des FPGAs aus, und löst einen Reset aus, der eine Rekonfiguration des FPGAs zu Folge hätte.

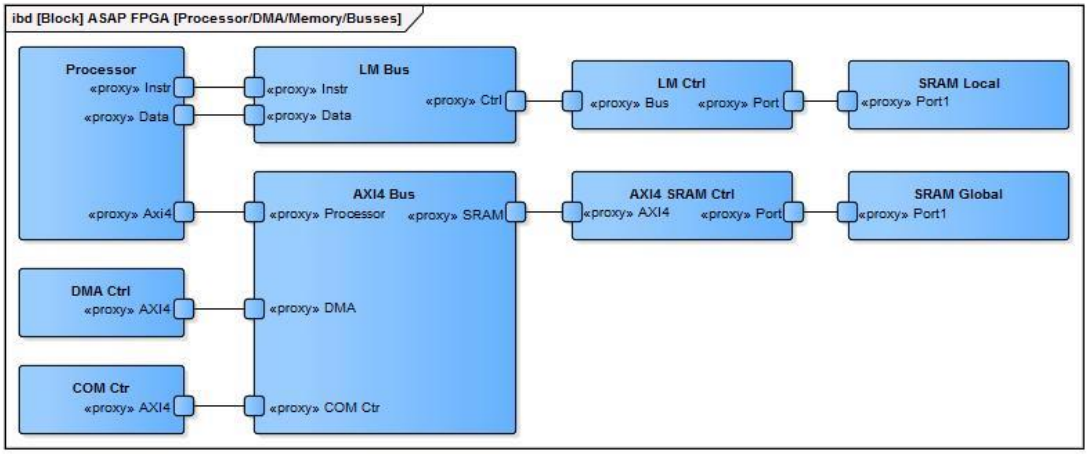


Abbildung 7-66: Prozessor, Speicher und Busse im FPGA

Als nächstes soll auf die im FPGA realisierten Grundkomponenten, die generell zur Ausführung der Software benötigt werden und die in der Lage sind, eigenständig Daten zu verschieben, eingegangen werden. Dazu gehören ein Prozessor, diverse Zugriffskomponenten auf Speicher und Busse. Die obige Abbildung 7-66 zeigt schematisch deren Verknüpfung untereinander.

Bei dem Prozessor handelt es sich um einen Microblaze des Herstellers Xilinx, der, da er in einem FPGA realisiert ist, individuell zusammengestellt wurde. Er weist unter anderem folgende Merkmale auf:

- Harvard-Architektur
- 80 MHz
- FPU (Floating Point Unit)
- BS (Barrel Shifter)

Der Prozessor ist an zwei verschiedene Datenbusse angeschlossen:

- LM-Bus (Local Memory)
- AXI-Bus (Advanced Extensible Interface)

Der Prozessor nutzt den LM-Bus für den Zugriff auf die eigentlichen Instruktionen und Daten, die sich in einem statischen Speicher (192 KByte) befinden. Sein Zugriff ist exklusiv (daher lokal). Zusätzlich kann er über den AXI-Bus auf einen weiteren statischen Speicher (128 KByte) zugreifen. Dieser ist jedoch global, da auch andere Teilnehmer am Bus, die sogenannte Peripherie, auf den Speicher zugreifen können. Der Datenaustausch mit den Speichern wird generell über Speicherkontroller (*LM Ctrl* und *AXI SRAM Ctrl*) gesteuert. Diese unterstützen den Error-Correction-Code (ECC), der zum Beheben von 1-Bit-Fehlern und zum Detektieren von 2-Bit-Fehlern im Speicher nützlich ist. Da bei in Weltallumgebung eingesetzter Elektronik mit erhöhten Wahrscheinlichkeiten für Single-Event-Upsets (SEU) und Single-Event-LatchUps (SEL) zu rechnen ist, wird der ECC standardmäßig bei ASAP aktiviert. Der ECC-Schutz prädestiniert diese Speicher vor allem für das auszuführende Programm und für kritische Daten. Neben dem Prozessor gibt es noch zwei weitere Komponenten am AXI-Bus, die aktiv Daten verschieben können. Erstens, den DMA-Controller (Direct Memory Access) und zweitens, den COM-Controller. Ersterer hat die Aufgabe, auf Anweisung des Prozessors, selbstständig Daten zwischen Speicherbereichen zu verschieben. Letzterer dient dazu, den Prozessor bei der externen Kommunikation zu entlasten. Dazu muss er diverse Protokolle implementieren. Alle anderen Komponenten bzw. Peripherie sind passiver Art und können nicht eigenmächtig einen Datenaustausch initiieren.

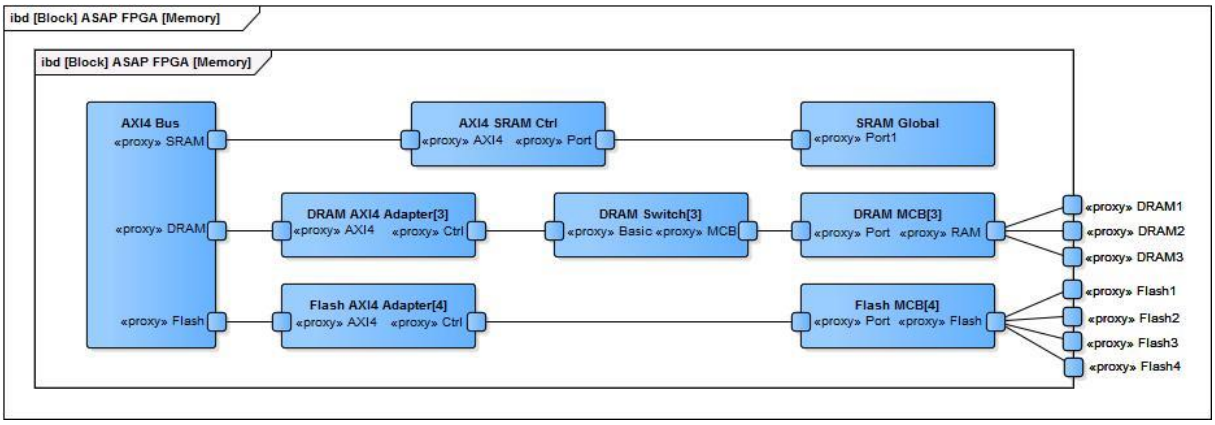


Abbildung 7-67: Arten von Speicher

Insgesamt gibt es vier verschiedene Arten von Speichern, mit denen die aktiven Komponenten, d.h. der Prozessor, der DMA- und der COM-Controller, agieren können. Der lokale und der globale SRAM wurden bereits vorgestellt. Beide Speicher werden durch die Ressourcen des FPGAs intern realisiert und weisen eine relativ kleine Kapazität im Bereich von KBytes auf. Die anderen zwei Arten von Speichern, DRAM und Flash, sind zwar extern, d.h. in anderen Gehäusen als dem des FPGAs untergebracht, werden aber mit Hilfe zusätzlicher Komponenten im FPGA an den AXI-Bus angeschlossen. Die obige Abbildung 7-67 zeigt die dafür notwendige Verschaltung.

DRAM- und Flash-ICs werden mit Hilfe von MCBs (Memory Controller Block) im FPGA angesteuert. Das sind spezielle Komponenten, die die Protokolle implementiert haben, die für den Datendatenaustausch mit diesen externen Speichern notwendig sind. Der Signalaustausch erfolgt über die entsprechenden IO-Schnittstellen des FPGAs. Da die MCBs keine Schnittstelle, die mit dem AXI4-Bus kompatibel ist, besitzen, müssen noch spezielle Adapter (*DRAM AXI4 Adapter* und *Flash AXI4 Adapter*) zwischengeschaltet werden, die die MCBs an den AXI4-Bus anbinden. Der Abbildung kann zusätzlich entnommen werden, dass bei den DRAMs noch Schaltkomponenten (DRAM Switch) eingebunden sind. Auf diese Thematik soll an späterer Stelle näher eingegangen werden.

Als nächstes soll die Kommunikation von ASAP betrachtet werden. Hierbei wird zwischen der externen und der internen Kommunikation unterschieden. Das Prinzip ähnelt dem bei den Speichern. Die externe Kommunikation ist jene, deren Pfade die Systemgrenzen von ASAP verlassen und somit an die Steckverbinder nach außen geführt sind. Sie ermöglicht den Datenaustausch mit anderen Komponenten im Satelliten, zum Beispiel dem OBC oder dem PDH. Die interne Kommunikation verbleibt innerhalb der Systemgrenzen und ist damit von außen nicht zugänglich. Zum Beispiel dient sie dem Auslesen der in ASAP verbauten Strom- und Temperatursensoren. Die nachfolgende Abbildung 7-68 zeigt eine Übersicht, wie der externe Datenaustausch realisiert wird. Dieser wird generell in zwei Gruppen von Datenbussen unterteilt:

- LS-Busse mit geringen Übertragungsraten bis 1Mbit/s (LS für Low-Speed)
- HS-Busse mit höheren Übertragungsraten über 1Mbit/s (HS für High-Speed)

Unabhängig von Art der Gruppe ist jeder Datenbus aus Sicherheitsgründen redundant (1 + 1) ausgeführt.

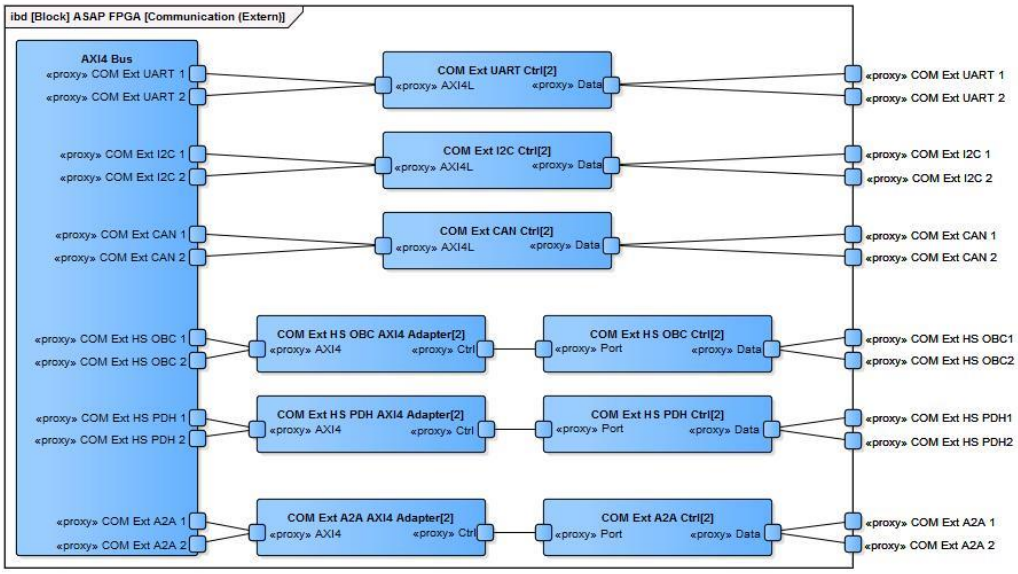


Abbildung 7-68: externe Kommunikation

Zu den LS-Bussen gehören der UART (Universal Asynchronous Receiver Transmitter), der I2C (Inter Integrated Circuit), der CAN (Controller Area Network) und der proprietäre A2A (für die Kommunikation der beiden Hauptplatinen untereinander). Für die HS-Datenübertragung steht zum OBC und zum PDH jeweils ein redundanter Bus bereit. Diese sind bewusst generisch gehalten und basieren auf jeweils 4 LVDS-Signalleitungspaaren (Low Voltage Differential Signaling). Die theoretisch maximale Bandbreite pro Paar beträgt 800Mbit/s. Der LVDS-Bus bildet die Ausgangslage für einen der folgenden möglichen Protokollstapel (Protocol Stacks):

- Proprietäres Protokoll
- SpaceWire
- Ethernet/IP/TCP (mit eigenem PHY-Layer basierend auf LVDS)

Es ist dem Anwender von ASAP überlassen, welche Busse eingesetzt werden sollen. Die Auswahl hängt vom zu realisierenden Szenario (siehe Abschnitt 7.1) ab.

Damit die Master (z.B. Prozessor, DMA) auf die Kommunikationsbusse (z.B. CAN) zugreifen können, müssen diese an den AXI4-Bus angeschlossen werden. Dafür sind die jeweiligen Controller (von Xilinx) zuständig, die die Protokolle der Kommunikationsbusse implementieren und eine entsprechende AXI4-Schnittstelle per Definition besitzen. Für die Controller der proprietären Busse müssen zusätzlich noch Adapter implementiert werden, die den Zugriff auf den AXI4-Bus ermöglichen.

Der Aufbau der internen Kommunikation erfolgt analog zur externen (siehe Abbildung 7-69), besteht jedoch nur aus einem einzigen I2C-Bus, an dem die Spannungs-/Strom- und Temperatursensoren angeschlossen sind.

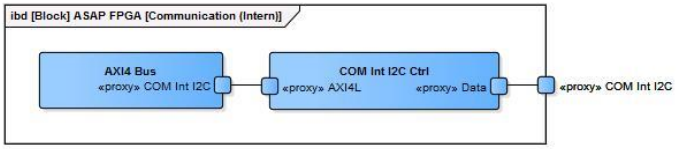


Abbildung 7-69: interne Kommunikation

Ein weiterer Aspekt der Schaltung betrifft die Ansteuerung der Bildsensoren zusammen mit der Zwischenspeicherung und der Auswertung von Bilddaten (siehe Abbildung 7-70).

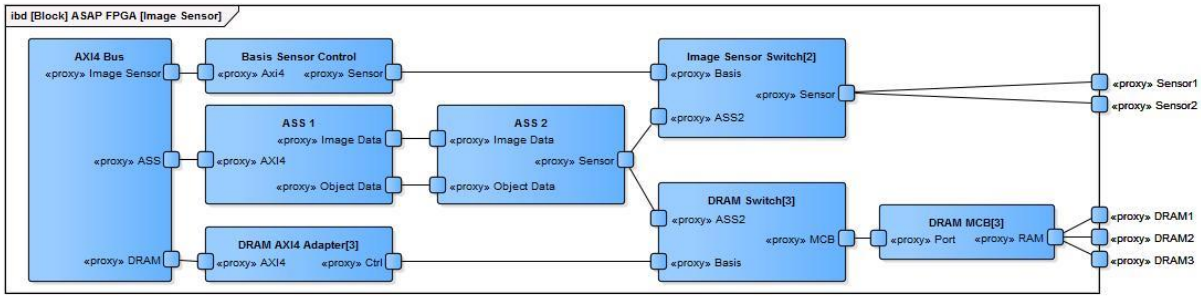


Abbildung 7-70: Ansteuerung der Bildsensoren

Hierbei tritt ein Konflikt auf. Es gibt zwei Komponenten, die Basissteuerung (Basis Sensor Control) und die autonome Steuerung (Autonomous Sensor System, ASS 2), die beide den Zugriff sowohl auf den Bildsensor als auch den DRAM benötigen. Die Basissteuerung führt routinemäßige Tests an Speicher und Bildsensoren durch, während das ASS die eigentlich geforderte autonome Detektion vornimmt. Zur Lösung des Konfliktes wurden Multiplexer (*Image Sensor Switch* und *DRAM Switch*) zwischengeschaltet, die eine Entkoppelung bewirken, indem sie nur einer Komponente den Zugriff gestatten. Die Schaltelemente werden je nach Betriebsart (Basis oder autonome Detektion) zuvor durch Software eingestellt.

Abgrenzung der Beiträge des Autors:

An der Verarbeitung der Bilddaten (siehe Abbildung 7-64) sind maßgeblich zwei Prozesse (*ASS1 Process* und *ASS2 Process*) beteiligt. Der Prozess ASS2, der aus dem Bildsensor die Rohdaten direkt ausliest und hieraus die ersten Meta-Daten generiert, wurde von Herrn Balagurin entwickelt. Für alle anderen in diesem Abschnitt vorgestellten Prozesse (einschließlich *ASS1 Process*) war der Autor verantwortlich.

7.3.5.3 Software

Die Software von ASAP besteht, wie schon im Abschnitt 7.3.5 erläutert wurde, aus drei verschiedenen Anwendungen, die jedoch eine Gemeinsamkeit aufweisen: alle drei sind in den Betrieb des Satelliten eingebunden und haben dementsprechend eine Schnittstelle zur Kommandierung und zur Telemetrierung. Die nachfolgende Abbildung 7-71 zeigt daher die gemeinsamen Funktionalitäten.

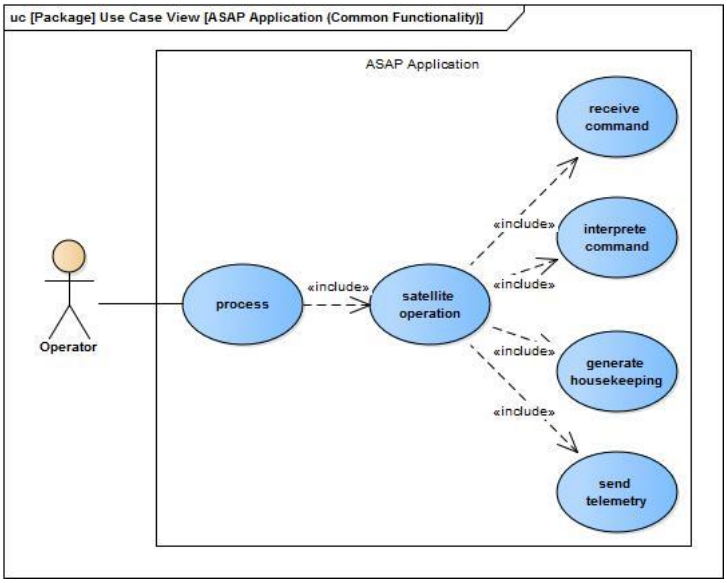


Abbildung 7-71: Gemeinsame Funktionalitäten der ASAP Anwendungen

Jede Anwendung hat einen Prozess (*process*) oder mehrere Prozesse abzuarbeiten. In der Regel muss der Operator auf diese Einfluss per Kommandierung und Telemetrie ausüben. Zu diesem Zweck gibt es einige gemeinsame Teilfunktionalitäten:

- Empfangen von Kommandos (*receive command*)
- Interpretieren von Kommandos (*interprete command*)
- Senden von Telemetrie (*send telemetry*)
- Bereitstellen von Zustandsdaten (*generate housekeeping*)

Jede Anwendung verfügt über zwei RODOS-Topics (Cmd und Hk), mit denen der dafür notwendige Datenaustausch erfolgen soll.

In den nachfolgenden Unterabschnitten sollen diese drei Anwendungen genauer erläutert werden.

7.3.5.3.1 Basis

Die Basis-Anwendung dient dazu, die Plattform (Hardware und Schaltkreis) von ASAP je nach Bedarf einzustellen und zu überwachen. Die unten stehende Abbildung 7-72 zeigt die dabei involvierten Funktionalitäten.

Eine grundlegende Aufgabe der Basis-Anwendung besteht darin, die Datenflüsse (siehe Abschnitt 7.3.5.2) mittels der Schaltelemente (*process switches*) für den DRAM und den Bildsensor einzustellen. Somit kann vorgegeben werden, ob die Basis-Anwendung bzw. der Basis-Schaltkreis oder die ASS-Anwendung bzw. der ASS2-Schaltkreis, den Zugriff erhalten. Es kann der Fall eintreten, dass der Operator den Bildsensor ohne autonome Funktionen, also als reine Kamera, auslesen möchte. Dazu muss eine direkte Steuerung (*process image sensor*) des Bildsensors ermöglicht werden. Des Weiteren sind auf der Hardware diverse andere Sensoren zur Zustandserfassung, z.B. Strom- und Temperatursensoren, verbaut. Auch diese müssen angesteuert und ausgelesen (*process*

misc sensor) werden. Die dabei gewonnenen Daten stellen einen Teil der Housekeeping vom ASAP-System dar. Ein weiterer Aspekt betrifft den Speicher. ASAP verfügt über größere Mengen an DRAM und Flash-Speicher. Es ist denkbar, dass der Operator im konventionellen Betrieb herkömmliche Speicheroperationen ausführen möchte. Dazu zählen unter anderem das Lesen, das Schreiben, das Löschen, das Verschieben und das Defragmentieren bestimmter Speicherbereiche. Diese Operationen (*process memory*) gilt es zu unterstützen. Dabei ist es aus Sicht des Operators irrelevant, ob es sich um Speicherbereiche im DRAM oder Flash handelt. Als letztes gilt es, diverse Testfunktionalitäten (*process tests*) für die Hardware und die Schaltkreise umzusetzen. Sie können in Problemfällen vom Operator aufgerufen werden, um Beschädigungen an der ASAP Plattform festzustellen.

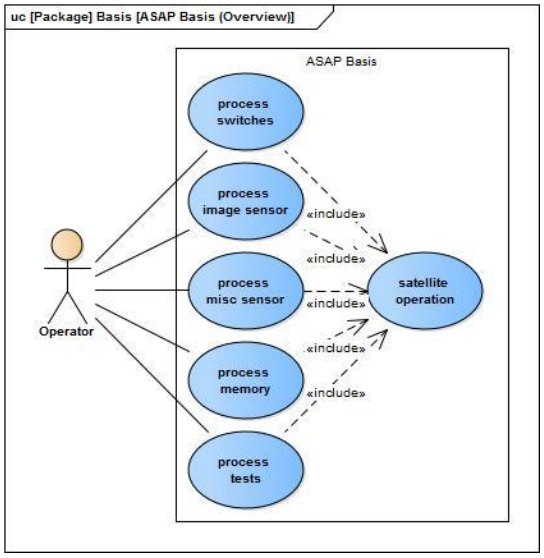


Abbildung 7-72: Funktionalitäten der Basis-Anwendung

7.3.5.3.2 Autonomes Bildsensorsystem

Das autonome Bildsensorsystem (ASS) hat die Aufgabe, die aus den Bildern der Kamera gewonnenen vorläufigen Meta-Daten aufzubereiten, in das Endformat umzuwandeln, und anschließend über das entsprechende RODOS-Topic an das autonome Planungssystem (APS) zu versenden. Die nachfolgende Abbildung 7-71 zeigt die dabei notwendigen Arbeitsschritte.

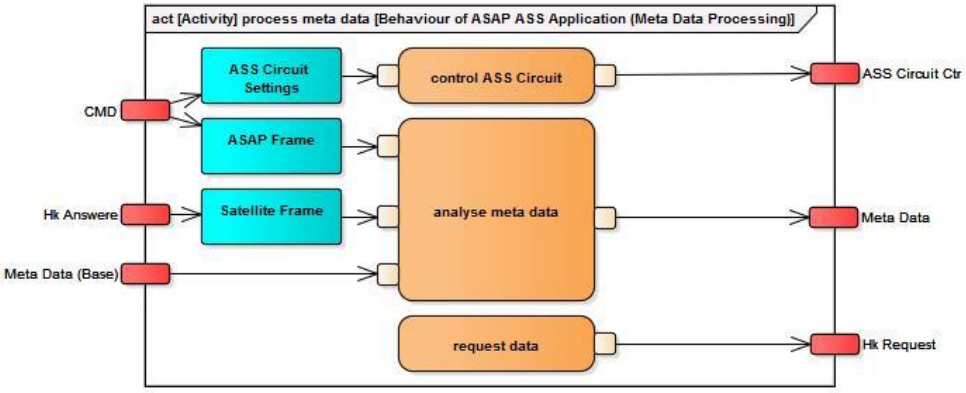


Abbildung 7-73: Abläufe bei der Aufbereitung der Meta-Daten

Die Endfassung der Meta-Daten (*Meta Data*) werden als Botschaften verschickt. Diese weisen zur einfacheren Handhabung durch das APS eine in der nachfolgenden Tabelle 7-15 beschriebene

Datenstruktur auf. Die Inhalte hängen vom Referenzszenario ab und können dementsprechend frei definiert werden.

Byte-Position	Name	Beschreibung
0 – 1	Key	Der Schlüssel ist das Identifizierungsmerkmal für jeden konkreten Gegenstand, auf den sich die Meta-Daten beziehen.
2 – 3	Type	Der Typ gibt die Art (Objekt, Phänomen, Ereignis) des Gegenstands an.
4	Action	Die Aktion gibt die Art des Auftretens an: 1 = Der Gegenstand wurde zum ersten Mal erfasst. 2 = Der Gegenstand wurde bereits entdeckt, die Informationen werden aber aktualisiert. 3 = Der Gegenstand wird nicht mehr erfasst.
5	Frame	Gibt das Bezugssystem für die Koordinaten an. 1 = Earth Centered Intertial (ECI) 2 = Earth Centered Fixed (ECF)
6 – 7	Length	Die Länge der zusätzlichen Daten
8 – 15	X – Koordinate	Die X – Koordinate bezogen auf das im Feld „Frame“ angegebene Bezugssystem
16 – 24	Y – Koordinate	Die Y – Koordinate bezogen auf das im Feld „Frame“ angegebene Bezugssystem
25 – 31	Z – Koordinate	Die Z – Koordinate bezogen auf das im Feld „Frame“ angegebene Bezugssystem
>= 32	Daten	Zusätzliche Informationen

Tabelle 7-15: Struktur der Meta-Daten (Endformat)

Das ASS arbeitet bei der Erstellung der Meta-Daten im Verbund mit einem Schaltkreis (im FPGA). Dieser wurde im Abschnitt 7.3.5.2 vorgestellt und liefert der Anwendung eine Ausgangsbasis der Meta-Daten (*Meta Data Base*). Der Schaltkreis kann an bestimmte Situationen angepasst (*control ASS circuit*) werden. Die dafür notwendigen Einstellungen (*ASS Circuit Settings*) können per Kommandierung über das RODOS-Topic *CMD* vorgenommen werden. Ausgangsseitig (*ASS Circuit Ctr*) ist der Prozessor mit dem betroffenen Schaltkreis verbunden.

Die bisher erfolgte Analyse der Meta-Daten (Base) durch den Schaltkreis ist in dem Sinne unvollständig, als dass noch kein Bezug zu einem dreidimensionalen Referenzsystem, z.B. dem ECI (Earth Centered Inertial) oder dem ECF (Earth Centered Fixed) hergestellt wurde. Das Lageregelungssystem des Satelliten benötigt aber als Vorgabe, z.B. zur Nachverfolgung von Zielen (siehe Abschnitt 7.1 - Referenzszenarien), deren Koordinaten in einem der zwei soeben genannten Referenzsysteme. Die in den bisherigen Meta-Daten (Base) angegebenen Koordinaten sind jedoch zweidimensional und beziehen sich lediglich auf die Ebene des Bildsensors. Eine im Rahmen der Analyse (*analyze meta data*) direkte Umwandlung der Koordinaten in das andere Bezugssystem mittels Transformation ist nicht möglich, da die vorliegenden Angaben (z.B. bezüglich der Optik) für eine Umwandlung zu unbestimmt sind. Allerdings kann mit Hilfe einer geometrischen Betrachtung[5] die Richtung des Zielobjekts aus dem Sichtbereich (FOV) bestimmt werden. Die nachfolgende Abbildung 7-74 veranschaulicht den Zusammenhang.

[5] Die Überlegungen und Konzepte zur Optik und Geometrie stammen von Herrn Balagurin, die Zeichnung sowie die nachfolgenden mathematischen Formeln wurden jedoch vom Autor angefertigt bzw. hergeleitet.

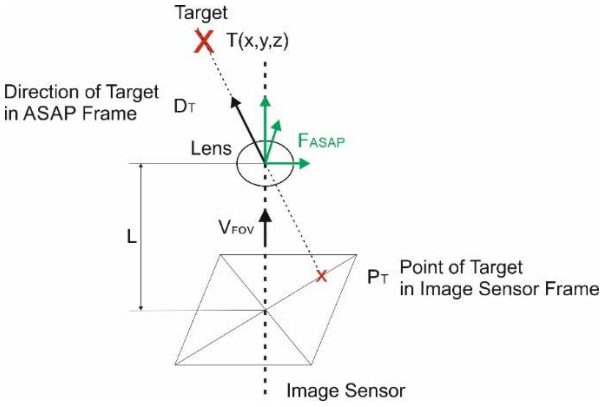


Abbildung 7-74: Bestimmung der Richtung des Zielobjekts

Die Ebene des Bildsensors und der dazu orthogonal stehende Richtungsvektor V_{FOV} des Sichtbereichs bilden zusammen für die Kamera, bzw. ASAP, ein Bezugssystem F_{ASAP}, das seinen Ursprung im Brennpunkt der Optik hat. Die Brennweite L ist der Abstand zwischen der Ebene des Bildsensors und dem Brennpunkt. Mit diesen Angaben kann die Richtung eines Objektes (*Target*) im Sichtfeld der Kamera mit Hilfe seiner Abbildung P_T auf dem Bildsensor bestimmt werden.

$$\overrightarrow{D_T_F_{ASAP}} = (P_T.X \,/\, P_T.Y \,/\, L)$$

Die Richtungsangabe bezieht sich jedoch auf das Bezugssystem F_{ASAP}. Dieses Ergebnis muss noch auf ein erdbezogenes Referenzsystem (z.B. ECI) transformiert werden, damit es als Teil der Meta-Daten verwendet werden kann. F_{ASAP} bezieht sich auf F_{SAT}, dem Bezugssystem des Satelliten, dieses wiederum bezieht sich auf F_{ECI}, dem ECI-Bezugssystem. Durch Kenntnis dieser Zusammenhänge kann die Richtung des Zielobjektes transformiert werden:

$$\overrightarrow{D_T_F_{ECI}} = {F_{SAT}}^{-1} * {F_{ASAP}}^{-1} * \overrightarrow{D_T_F_{ASAP}}$$

F_{ASAP} wird mittels Kommandierung vorgegeben (*ASAP Frame*) und bleibt statisch. F_{SAT} ist dynamisch und ändert sich laufend, zum einen auf Grund der Bahnbewegung des Satelliten, und zum anderen auf Grund des aktiven Eingriffs des Lageregelungssystems. Die ASS Anwendung kann von der Anwendung zur Erfassung der Zustandsdaten (*Housekeeping Application*) den aktuellen Wert (*Satellite Frame*) von F_{SAT} anfordern (*request data*).

7.3.5.3.3 Autonomes Planungssystem

Das autonome Planungssystems (APS) wird als eine Anwendung in ASAP eingesetzt, um den autonomen Betrieb des Satelliten zu ermöglichen. Seine Funktionsweise wurde in Abschnitt 7.2 detailliert beschrieben. Ein wichtiger Aspekt ist dabei seine Konfiguration. Sie legt im Wesentlichen das autonome Verhalten fest. Dabei werden sogenannte Definitionen für die verschiedenen Aspekte (z.B. Aktivitäten, Ressourcen, Profile, …) der autonomen Planung gesetzt. In diesem Abschnitt soll eine Konfiguration, die sich an die im Abschnitt 7.1 beschriebenen Referenzszenarien und den Satelliten SONATE als Zielplattform orientiert, veranschaulicht werden. Zum Zeitpunkt der Fertigstellung der vorliegenden Dissertation war die Entwurfsphase für SONATE noch nicht abgeschlossen. Deswegen muss die in diesem Abschnitt beschriebene Konfiguration als eine vorläufige Version der Konfiguration betrachtet werden.

Als erstes sollen die Ressourcen betrachtet werden:

Name	Wertebereich		Beschreibung
ACS	0 – 1	[1]	Zugriff auf das Lageregelungs-Subsystem des Satelliten
COM	0 – 1	[1]	Zugriff auf das Kommunikations-Subsystem des Satelliten

ASAP Camera	0 – 1	[1]	Zugriff auf das optische System von ASAP
ASAP Memory DRAM	0 – 1024	[MB]	Der zur Verfügung stehende flüchtige Speicher für die Nutzlasten
ASAP Memory Flash	0 – 16384	[MB]	Der zur Verfügung stehende nichtflüchtige Speicher für die Nutzlasten
Electric Power	0 – (*)	[W]	Die zur Verfügung stehende elektrische Leistung im Satelliten
Electric Energy	0 – (*)	[Ah]	Die zur Verfügung stehende elektrische Energie im Satelliten

Tabelle 7-16: Ressourcen der autonomen Planung

(*) Zum Zeitpunkt der Fertigstellung der vorliegenden Dissertation konnten keine Angaben zu den Werten gegeben werden

Als nächstes werden die in der autonomen Planung berücksichtigten Aktivitäten gelistet:

Name	**Beschreibung**
Idle	Der Satellit ist inaktiv und führt keine besondere Tätigkeit aus.
Search	Der Satellit beobachtet mit der ASAP Kamera den Horizont der Erde und versucht, besondere Ereignisse und Phänomene zu detektieren. Die Lage des Satelliten wird dabei so geregelt, dass der Horizont immer einen bestimmten Teil des Sichtfeldes der Kamera ausfüllt.
Track	Der Satellit versucht, die Kamera für einen bestimmten Zeitraum auf ein bestimmtes Ziel auszurichten.
Image	Der Satellit versucht, die Kamera auf ein bestimmtes Ziel auszurichten und eine Momentaufnahme durchzuführen.
Video	Der Satellit führt in regelmäßigen Zeitabständen Momentaufnahmen durch. Die Ausrichtung des Satelliten soll dabei bei Bedarf kontrolliert werden.
COM Downlink LS	Der Satellit führt einen Kommunikationspass mit der Bodenstation für niedrige Bandbreiten durch. Die Ausrichtung des Satelliten muss nicht kontrolliert werden.
COM Downlink HS	Der Satellit führt einen Kommunikationspass mit der Bodenstation für höhere Bandbreiten durch. Die Ausrichtung des Satelliten mittels Lageregelung muss kontrolliert werden.
Recharge	Der Satellit richtet seine Solarpanelle in Richtung Sonne aus, um seine Energiespeicher wieder aufzuladen.

Tabelle 7-17: Aktivitäten der autonomen Planung

Im Folgenden werden einige Profile mit Hilfe von Abbildungen vorgestellt. Mit einem Profil lässt sich die Auswirkung einer Aktivität auf eine Ressource modellieren. Der zeitliche Verlauf ergibt sich aus Zustandsübergängen (S=Start, P=Pause, C=Continue, F=Finish) der jeweiligen Aktivität.

Die nachfolgende Abbildung 7-75 zeigt das Profil für einen exklusiven Zugriff auf eine Ressource.

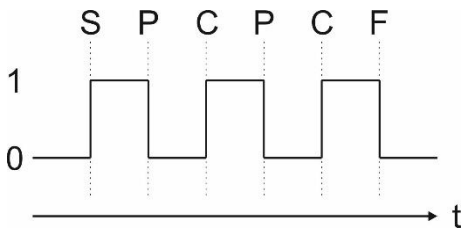


Abbildung 7-75: Profil für exklusiven Zugriff auf eine Ressource

Es wird verwendet, um festzulegen, dass auf eine Ressource (z.B. ACS, COM) zur selben Zeit nur von einer Aktivität (z.B. Search, Image, Downlink) aus zugegriffen werden kann. Hierbei wird vorausgesetzt, dass die betroffene Ressource nur die Werte „0“ (frei) und „1“ (belegt) annehmen kann.

Die nachfolgende Abbildung 7-76 zeigt das Profil für eine statische Auswirkung auf eine Ressource.

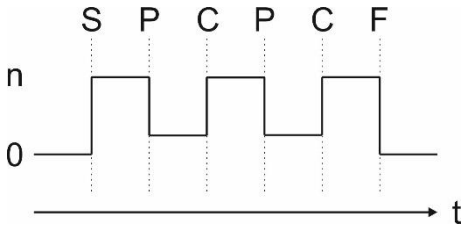


Abbildung 7-76: Profil für eine statische Auswirkung auf eine Ressource

Diese Art von Profil wird eingesetzt, um zum Beispiel den elektrischen Leistungsbedarf von Aktivitäten zu modellieren. In dem aktuell vorliegenden konkreten Fall benötigt eine Aktivität (z.B. Search) elektrische Leistung, obwohl sie sich im inaktiven Zustand (Pause) befindet. Damit kann die Tatsache modelliert werden, dass zum Beispiel der DRAM von ASAP, obwohl keine Zugriffe erfolgen, Leistung verbraucht, um die Informationen in den Speicherzellen zu erhalten (Refresh-Zyklen).

Die nachfolgende Abbildung 7-77 zeigt das Profil für eine integrierende Auswirkung auf eine Ressource.

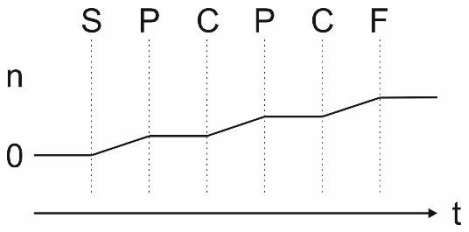


Abbildung 7-77: Profil für integrierende Auswirkung auf eine Ressource

Dieses Verhalten tritt besonders beim Verbrauch von Speicher durch Aktivitäten auf. Im passiven Zustand bleibt die Ressource statisch, im aktiven Zustand nimmt der Wert der Ressource linear über die Zeit zu.

Zur Realisierung der in Abschnitt 7.1 dargestellten Referenzszenarien benötigt das APS zur autonomen Planung spezielle Informationen bzgl. der Raumkoordinaten verschiedener Objekte (z.B. Sonne, Mond, Satellit, Bodenstation) zu einem bestimmten Zeitpunkt. Diese Daten werden durch das autonome Planungssystem (siehe Abschnitt 7.2) nicht erzeugt und müssen daher durch einen spezialisierten Softwaredienst (siehe Abschnitt 7.2.6.1) generiert und als Meta-Daten ausgegeben werden. Zu diesem Zweck wurde für die Szenarien ein eigener Orbit-Dienst erstellt. In diesem können, je nach Konfiguration, ein oder mehrere Bahnmodelle (z.B. Kepler, SGP4), die die Aufenthaltsorte von Objekten im Raum zu einer bestimmten Zeit berechnen können, hinterlegt werden. Insgesamt verfügt der Dienst über drei verschiedene abgeleitete und implementierte Funktionalitäten/Verhaltensweisen (RODOS Job Behaviour):

1. Positionsbestimmung von Objekten auf Umlaufbahnen
   Der Orbit-Dienst berechnet auf Anfrage die Ortskoordinaten eines Objektes auf seiner Umlaufbahn zu einer bestimmten Zeit. Das Ergebnis ist ein einzelner Ortsvektor des Zielobjekts.

2. Sichtbarkeitsbereich von Objekten auf der Erde (siehe Abbildung 7-78)
   Der Orbit-Dienst berechnet auf Anfrage für ein Zeitintervall, wann ein Quellobjekt (z.B. Satellit) auf einer Umlaufbahn in den Sichtbarkeitsbereich eines Zielobjekts (z.B. Bodenstation) auf der Erde eintritt. Dabei werden jeweils die Zeitpunkte und die Ortskoordinaten zum Ein- und Austritt berechnet. Der Bereich der Sichtbarkeit wird durch Elevationswinkel des Zielobjekts eingeschränkt. Dabei muss berücksichtigt werden, dass durch die Drehbewegung der Erde sich der Aufenthaltsort (in ECI-Koordinaten) des Zielobjekts über die Zeit ändert. Das

Ergebnis ist eine Sequenz von Raum- und Zeitangaben entsprechend den berechneten Sichtbarkeitsbereichen im vorgegebenen Zeitintervall.

3. Sichtbarkeitsbereich von Objekten auf Umlaufbahnen (siehe Abbildung 7-79)
   Der Orbit-Dienst berechnet auf Anfrage für ein Zeitintervall, wann ein Quellobjekt (z.B. Satellit: SONATE) auf einer Umlaufbahn in den Sichtbarkeitsbereich eines Zielobjekts (z.B. Station: ISS) eintritt. Dabei werden jeweils die Zeitpunkte und die Ortskoordinaten zum Ein- und Austritt berechnet. Der Bereich der Sichtbarkeit wird durch eine maximal zulässige Entfernung eingeschränkt. Dabei muss berücksichtigt werden, dass durch die Bewegung sowohl des Quellobjekts als auch des Zielobjekts der Sichtbarkeitsbereich über die Zeit wandert. Das Ergebnis ist eine Sequenz von Raum- und Zeitangaben entsprechend den berechneten Sichtbarkeitsbereichen im vorgegebenen Zeitintervall.

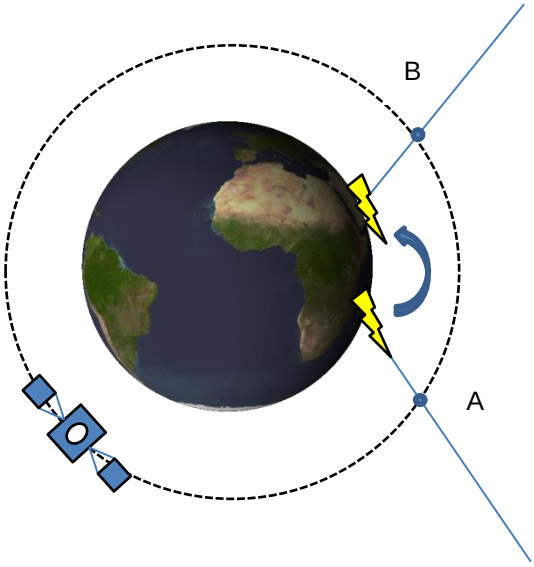


Abbildung 7-78: Sichtbarkeitsbereich von Objekten auf der Erde

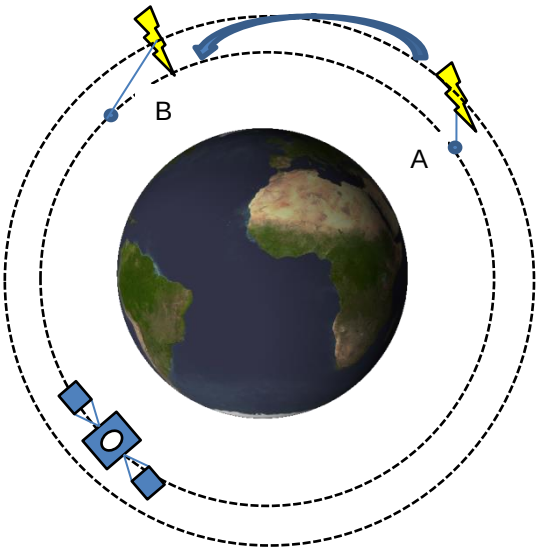


Abbildung 7-79: Sichtbarkeitsbereich von Objekten auf Umlaufbahnen

Das in den Referenzszenarien beschriebene Verhalten des Satelliten wird mit Hilfe von Entscheidern (siehe Abschnitt 7.2.5.4) umgesetzt. Konkret werden zwei Entscheider verschiedenen Typs eingesetzt:

Der erste Entscheider übernimmt direkte Anweisungen vom Operator:

1. Der Operator erzeugt und wartet Instanzen der Aktivität *Search*. Damit wird eingestellt, wann ASAP generell den Horizont der Erde nach Phänomenen absuchen kann.
2. Der Operator erzeugt und wartet Instanzen der Aktivität *Image*. Damit lassen sich Momentaufnahmen von bestimmten Zielen mit der Kamera erzeugen.
3. Der Operator erzeugt und wartet Instanzen der Aktivität *Record.* Damit lassen sich Sequenzen von Momentaufnahmen mit der Kamera erzeugen.

Der zweite Entscheider ist vom Typ her regelbasiert und hat folgende Regeln definiert:

1. Erzeuge mit jedem Sichtbarkeitsbereich zur Bodenstation (Meta-Datum von Orbit-Dienst) die Aktivitäten *Communicate LS* und *Communicate HS.*
2. Erzeuge mit jedem Sichtbarkeitsbereich zur Sonne (Meta-Datum von Orbit-Dienst) die Aktivität *Recharge.*

3. Erzeuge für jedes langfristige Phänomen (Meta-Datum von ASAP ASS) auf der Erde (z.B. Vulkanausbruch) die Aktivität *Track,* um eine Nachverfolgung durchzuführen. Übermittle die Daten an den Orbit-Dienst, um weitere Sichtbarkeitsbereiche zu berechnen.
4. Erzeuge mit jedem Sichtbarkeitsbereich zu einem langfristigen Phänomen (Meta-Datum von Orbit-Dienst) die Aktivität *Track,* um eine Nachverfolgung durchzuführen.

An dieser Stelle soll auf die Tatsache hingewiesen werden, dass das Erzeugen einer Aktivität (als Entscheidung) durch einen Entscheider nicht automatisch deren Ausführung bewirkt. Wie bereits ausführlich geschildert wurde, findet gleichzeitig zur Entscheidungsfindung auch die Ablaufplanung und Planausführung statt. Die Aktivitäten *Search*, *Image* und *Record* generieren zum Beispiel Nutzlastdaten und belegen entsprechend den Speicher *ASAP Memory*. Die Aktivität *Communication HS* entlastet Speicher durch einen Download. Ob eine Gelegenheit genutzt wird, hängt vom Ablaufplaner ab.

7.4 ATUS – Testumgebungssystem

In den vorherigen Abschnitten wurden sowohl das autonome Planungssystem APS als auch die autonome Plattform ASAP beschrieben. Beide Systeme sind hochkomplex. Daher müssen deren Funktionalitäten auf Korrektheit mit Hilfe eingehender Tests verifiziert werden. Zu diesem Zweck wird im Rahmen des Vorhabens ATUS (ASAP/APS Testumgebungssystem) eine Testumgebung entwickelt, die sowohl als ein Hardware-In-The-Loop (HIL) Testsystem für ASAP als auch als ein Software-In-The-Loop (SIL) Testsystem für APS zu betrachten ist. Ein HIL emuliert für die zu testende Hardware (z.B. ASAP) eine Systemumgebung. Ein SIL wird verwendet, um eine Software zu testen, die noch nicht auf ihre vorgesehene Zielhardware (z.B. das APS auf ASAP) portiert wurde. Sie eignet sich daher besonders für das „Rapid Prototyping“. Das APS zum Beispiel ist auf diese Weise entstanden. Es wurde zuerst in C# für die Windows-Plattform geschrieben und im SIL getestet. Derzeit wird der Code von APS für die ASAP Plattform portiert.

Die Komponenten des Testumgebungssystems bestehen deswegen sowohl aus Hardware (siehe Abbildung 7-80) als auch aus Software (siehe Abbildung 7-81).

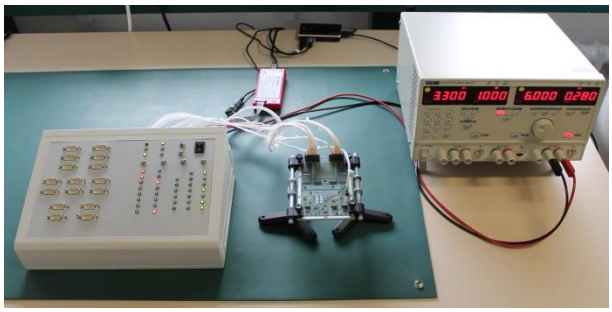


Abbildung 7-80: Hardware-Testumgebung (HIL/ Aufbau)

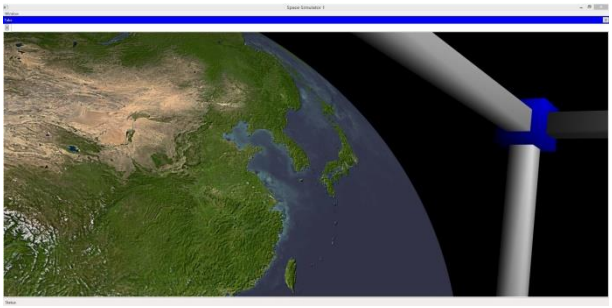


Abbildung 7-81: Software-Testumgebung (SIL / 3D-Visualisierung)

Die nachfolgenden Unterabschnitte beschreiben die einzelnen Schritte des Entwicklungsprozesses, angefangen bei der Zielstellung bis hin zu den Ergebnissen der Entwurfssynthese. Hierbei werden Hardware- und Softwarekomponenten getrennt voneinander behandelt.

7.4.1 Zielstellung

Die Hauptaufgabe von ATUS besteht darin, zu Testzwecken für die beiden autonomen Systeme ASAP (siehe Abschnitt 7.3) und APS (siehe Abschnitt 7.2) die Systemumgebung zu emulieren. Das APS ist eine reine Softwareanwendung, die in ASAP als Teilmodul eingesetzt wird. ASAP hingegen ist ein aus Hard- und Software bestehendes System. Dementsprechend muss ATUS eine mehrschichtige Systemumgebung implementieren. Diese besteht für ASAP zum einem aus bestimmten Subsystemen des Satelliten, die über elektrische Anschlüsse (Power, CTR und COM) mit ASAP verbunden werden, und zum anderen aus dem realen Raum und seiner elektromagnetischen Strahlung im visuellen Bereich für das Kamerasystem. Für das APS besteht die Systemumgebung typischerweise aus den Softwareanwendungen zur Verwaltung der Kommandos, der Zustandsdaten (Housekeeping) und der Meta-Daten im Satelliten.

Mit ATUS sollen sich verschiedene Testvarianten realisieren lassen, um die Tests an die unterschiedlichen zu durchlaufenden Entwicklungsstände der Testobjekte anpassen zu können. Die nachfolgende Abbildung 7-82 verdeutlicht diese Thematik. Mit dem SIL, dem HIL und ASAP stehen verschiedene Plattformen zur Verfügung, auf denen die Softwareanwendungen ausgeführt werden können. Zu Beginn der Entwicklung werden die Anwendungen für den SIL entworfen. Das hat den Vorteil, dass grundsätzliche Funktionalitäten bereits getestet werden können, auch wenn die

eigentlichen Zielplattformen noch nicht zur Verfügung stehen. Mit dem Fortschreiten der Projekte werden die Anwendungen portiert.

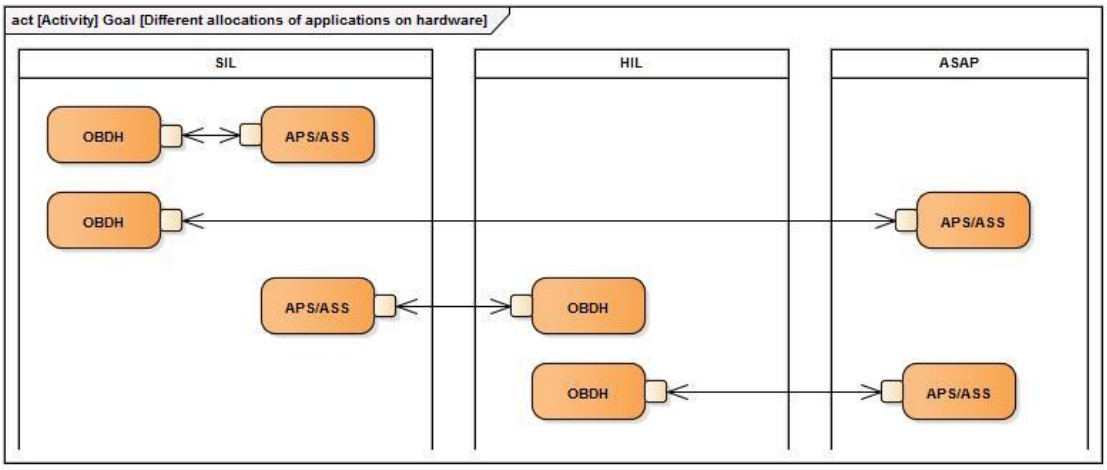


Abbildung 7-82: Verschiedene Varianten von Testaufbauten

Der HIL soll die Systemumgebung auf zwei verschiedene Arten simulieren können. Bereits entwickelte Subsysteme (z.B. OBC) des Satelliten können entweder direkt an das Testsystem angeschlossen werden. Dieses funktioniert in diesem Fall lediglich als Brücke, kann aber immerhin noch den Datenverkehr überwachen. Ansonsten muss das Testsystem durch eigene interne Komponenten die benötigten Funktionalitäten bereitstellen.

Der Fokus des Testsystems soll auf dem autonomen Betrieb liegen. Es soll gezeigt werden, dass ASAP/APS in der Lage ist, einen Satelliten autonom zu steuern. ATUS soll wie das APS so generisch sein, dass es für verschiedenen Szenarien (d.h. simulierte Satellitenmissionen) eingesetzt werden kann, es muss jedoch mindestens die in Abschnitt 7.1 skizzierten Referenzszenarien unterstützen.

7.4.2 Stakeholder

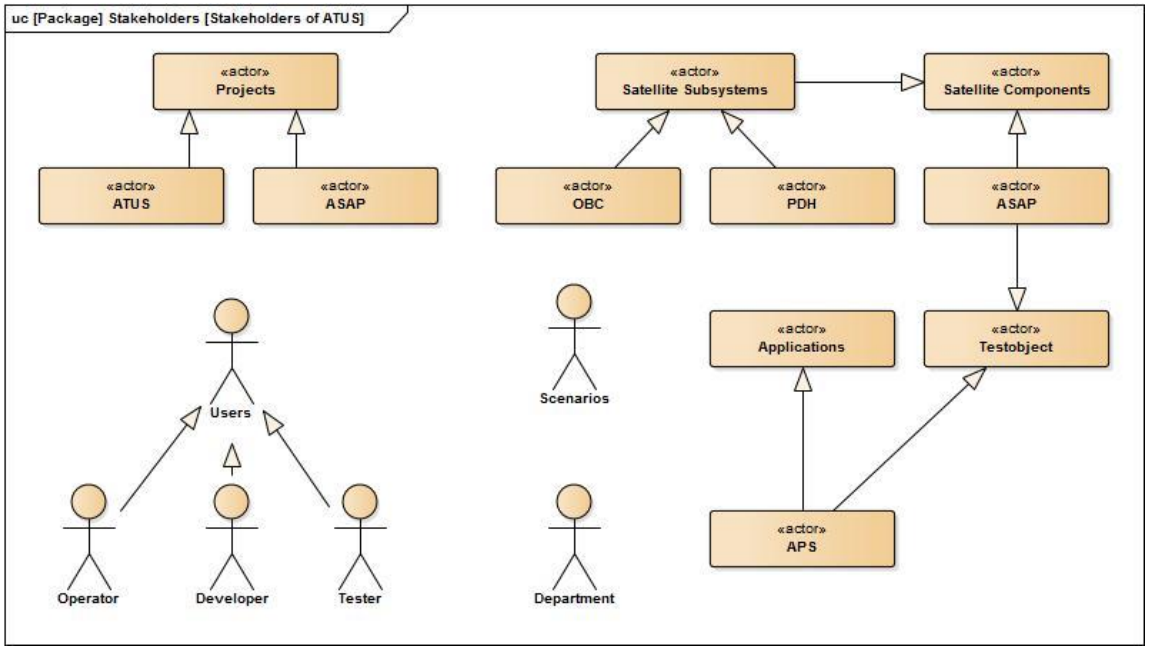


Abbildung 7-83: Stakeholder von ATUS

Die obige Abbildung 7-83 stellt eine Übersicht über alle Stakeholder dar, die eine relevante Beziehung zu ATUS haben.

Alle Stakeholder, das Testobjekt ausgeschlossen, wurden schon in den Abschnitten 7.2.2 und 7.3.2 bei der Beschreibung des autonomen Planungssystems APS bzw. von ASAP vorgestellt. Aus diesem Grund wird an dieser Stelle nicht näher auf die Stakeholder eingegangen.

7.4.3 Anforderungen

Eine Analyse der Beziehungen und Abhängigkeiten zwischen dem zu entwickelnden ATUS System und seinen Stakeholdern hat einen Satz von Nutzer- und Systemanforderungen, untergliedert in funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen, aufgezeigt. Sie werden in den nachfolgenden Tabellen aufgeführt. Die Anforderungen lassen sich zu inhaltlichen Gruppen, die jeweils einem Stakeholder zugeordnet sind, zusammenfassen. Elementare Anforderungen werden nach Möglichkeit begründet.

Die nachfolgende Tabelle 7-18 listet die Nutzeranforderungen an das ASAP System:

ID	Name und Begründung
1	Nutzeranforderungen
1.1	Anforderungen an das SIL
1.1.1	Das SIL muss dem Tester Steuerungsfunktionalitäten der Simulation bzgl. des Ablaufs (z.B. Start, Pause, Weiter, Stopp, Reset) anbieten.
1.1.2	Das SIL muss dem Tester Steuerungsfunktionalitäten der Simulation bzgl. der Zeit (Echtzeit, Zeitraffer) anbieten.
1.1.3	Das SIL muss dem Tester ermöglichen, seine Sitzung (Konfiguration und im Laufe der Simulation generierte Daten) nicht-flüchtig zu speichern und zu einem späteren Zeitpunkt wieder zu laden.
1.1.4	Das SIL muss dem Tester eine Sicht der Daten in tabellarischer Form, und wenn möglich in graphischer Form darstellen.
1.1.5	Das SIL muss die APS – Anwendung simulieren.
1.1.6	Das SIL muss den Kontext (z.B. Anwendung zur Kommandoverwaltung) der APS – Anwendung simulieren.
1.1.7	Das SIL muss einen (oder mehrere) Gateway(s) (z.B. für UART) simulieren.
1.1.8	Das SIL muss für die APS – Anwendung Einflüsse entsprechend den Referenzszenarien generieren.
1.1.9	Das SIL soll die simulierte Szene in einer 3D – Ansicht darstellen.
1.2	Anforderungen an das HIL
1.2.1	Das HIL muss dem Tester Steuerungsfunktionalitäten der Simulation bzgl. des Ablaufs (z.B. Start, Pause, Weiter, Stopp, Reset) anbieten.
1.2.2	Das HIL muss dem Tester Steuerungsfunktionalitäten der Simulation bzgl. der Zeit (Echtzeit, Zeitraffer) anbieten.
1.2.3	Das HIL muss die elektrische Schnittstelle von ASAP implementieren.
1.2.4	Das HIL sollte die optische Schnittstelle von ASAP implementieren und eine optische Simulation anbieten.
1.2.5	Das HIL muss ASAP mit verschiedenen Interaktionspartnern (z.B. OBC oder Testschaltung) verschalten (multiplexen).
1.2.6	Das HIL muss für ASAP unabhängig von den eigentlichen Satellitenbuskomponenten eigene Testschaltungen für die ASAP – Schnittstelle zur Verfügung stellen.
1.2.7	Das HIL soll es dem Tester ermöglichen, die Signale der elektrischen Schnittstelle über ein Oszilloskop abgreifen zu können.

Tabelle 7-18: Nutzeranforderungen an das ATUS - System

Die nachfolgende Tabelle 7-19 listet die funktionalen Anforderungen an das ATUS System

ID	Name und Begründung
2	Systemanforderungen (funktionale Anforderungen)
2.1	Anforderungen an das SIL
2.1.1	Anforderungen an das SIL seitens des APS als Testobjekt (Stakeholder)
2.1.1.1	Schnittstellen und Interaktionen
2.1.1.1.1	RODOS–Middleware (Protokoll auf Kommunikationsebene) Das SIL muss zur Kommunikation mit dem APS eine RODOS-Middleware implementieren.
2.1.1.1.2	Interaktion (Protokoll auf Anwendungsebene) Das SIL muss zur Interaktion mit dem APS dessen Anwendungsprotokoll einhalten.
2.1.2	Anforderungen an das SIL seitens des Testers (Stakeholder)
2.1.2.1	Operationen
2.1.2.1.1	Steuerung
2.1.2.1.1.1	Start Das SIL muss dem Anwender eine Möglichkeit bieten, den Simulator zu starten und damit den Zeitfluss zu beginnen.
2.1.2.1.1.2	Pause Das SIL muss dem Anwender eine Möglichkeit bieten, den Simulator zu pausieren und damit den Zeitfluss anzuhalten.
2.1.2.1.1.3	Weiter Das SIL muss dem Anwender eine Möglichkeit bieten, den Simulator nach einer Pause wieder zu starten und damit den Zeitfluss wieder laufen zu lassen.
2.1.2.1.1.4	Stopp Das SIL muss dem Anwender eine Möglichkeit bieten, den Simulator zu stoppen und damit den Zeitfluss endgültig anzuhalten.
2.1.2.1.1.5	Reset Das SIL muss dem Anwender eine Möglichkeit bieten, den Simulator wieder in den ursprünglichen Ausgangszustand zurück zu versetzen.
2.1.2.1.1.6	Zeit (Reset) einstellen Das SIL muss dem Anwender eine Möglichkeit bieten, den Zeitpunkt für einen Reset einzustellen.
2.1.2.1.1.7	Zeit – Modus Das SIL muss dem Anwender eine Möglichkeit bieten, den Zeit-Modus einzustellen.
2.1.2.1.1.8	Echtzeit – Modus Das SIL muss einen Zeitfluss, der der Echtzeit entspricht, unterstützen.
2.1.2.1.1.9	Zeitraffer – Modus Das SIL muss einen Zeitfluss, der gegenüber der Echtzeit skaliert ist, unterstützen.
2.1.2.2	Einstellungen
2.1.2.2.1	Speichern Das SIL muss dem Anwender eine Möglichkeit bieten, seine Einstellungen zu speichern.
2.1.2.2.2	Laden Das SIL muss dem Anwender eine Möglichkeit bieten, seine Einstellungen zu laden.
2.1.2.3	Datenanalyse
2.1.2.3.1	Tabelle Das SIL muss dem Anwender eine Möglichkeit bieten, die Daten der zu simulierenden Komponenten in tabellarischer Form einzusehen.
2.1.2.3.2	Graph Das SIL muss dem Anwender eine Möglichkeit bieten, die Daten der zu simulierenden Komponenten in Form eines Graphen darzustellen.
2.1.2.4	Datenvisualisierung
2.1.2.4.1	3D Darstellung Das SIL muss eine simulierte Szene aus der Sicht eines Betrachters (Kamera)

	in einer 3D-Ansicht darstellen können.
2.1.3	Anforderungen an das SIL seitens der Referenzszenarien (Stakeholder)
2.1.3.1	APS – Anwendung (autonomes Planungssystem) Das SIL muss die APS - Anwendung simulieren.
2.1.3.2	Kommandoverwaltungs-Anwendung des Bodensegments Das SIL muss die Kommandoverwaltungs-Anwendung des Bodensegments simulieren.
2.1.3.3	Telemetrieverwaltungs-Anwendung des Bodensegments Das SIL muss die Telemetrieverwaltungs-Anwendung des Bodensegments simulieren.
2.1.3.4	Kommunikations-Anwendung des Bodensegments Das SIL muss die Kommunikations-Anwendung des Bodensegments simulieren.
2.1.3.5	Kommandoverwaltungs-Anwendung des Raumsegments Das SIL muss die Kommandoverwaltungs-Anwendung des Raumsegments simulieren.
2.1.3.6	Telemetrieverwaltungs-Anwendung des Raumsegments Das SIL muss die Telemetrieverwaltungs-Anwendung des Raumsegments simulieren.
2.1.3.7	Kommunikations-Anwendung des Raumsegments Das SIL muss die Kommunikations-Anwendung des Raumsegments simulieren.
2.1.3.8	Energieversorgungs-Anwendung des Raumsegments Das SIL muss die Energieversorgungs-Anwendung des Raumsegments simulieren.
2.1.3.9	Lageregelungs-Anwendung des Raumsegments Das SIL muss die Lageregelungs-Anwendung des Raumsegments simulieren.
2.1.3.10	Orbit-Dienst des Raumsegments Das SIL muss den Orbit-Dienst des Raumsegments simulieren.
2.1.3.11	Weltraumumgebungssimulation Das SIL muss ein Modell des Weltraums simulieren.
2.1.3.11.1	Bahnberechnungen Das SIL muss Bahnen von Objekten im Weltraum berechnen und simulieren können.
2.1.4	Anforderungen an das SIL seitens der Satelliten-Komponenten (Stakeholder)
2.1.4.1	Gateway-Anwendung (RODOS) Das SIL muss eine Gateway-Anwendung (RODOS) simulieren.
2.1.4.2	HAL UART-Bus (RODOS) Das SIL muss eine Hardwareabstraktion (RODOS) für den UART-Bus implementieren.
2.1.4.3	HAL I2C-Bus (RODOS) Das SIL muss eine Hardwareabstraktion (RODOS) für den I2C-Bus implementieren.
2.1.4.4	HAL CAN-Bus (RODOS) Das SIL muss eine Hardwareabstraktion (RODOS) für den CAN-Bus implementieren.
2.1.4.5	HAL HS-Bus (RODOS) Das SIL muss eine Hardwareabstraktion (RODOS) für den HS-Bus von ASAP implementieren. Folgende Busse sind zu berücksichtigen: - Proprietärer Bus - Ethernet mit UDP oder TCP (mit LVDS-Signalen auf dem physikalischen Layer) - SpaceWire
2.2	Anforderungen an das HIL
2.2.1	Anforderungen an das HIL seitens ASAP (als Testobjekt)
2.2.1.1	Schnittstellen
2.2.1.1.1	Energieversorgung Das HIL muss ASAP mit elektrischer Energie versorgen.
2.2.1.1.2	Digitale IO-Leitungen Das HIL muss für ASAP digitale IO-Leitungen bereitstellen.
2.2.1.1.3	UART-Bus Das HIL muss für ASAP UART-Busse bereitstellen.

2.2.1.1.4	I2C-Bus Das HIL muss für ASAP I2C-Busse bereitstellen.
2.2.1.1.5	CAN-Bus Das HIL muss für ASAP CAN-Busse bereitstellen.
2.2.1.1.6	HS – Bus Das HIL muss für ASAP HS-Busse bereitstellen. Folgende Busse sind zu berücksichtigen: - Proprietärer Bus - Ethernet mit UDP oder TCP (mit LVDS-Signalen auf dem physikalischen Layer) - SpaceWire
2.2.1.1.7	Optik Das HIL muss für ASAP einen optischen Input generieren.
2.2.2	Anforderungen an das HIL seitens des Testers (Stakeholder)
2.2.2.1	Schaltfunktionalität Das HIL muss die Funktion als Schalter übernehmen und Signalgruppen von ASAP wahlweise mit Interaktionspartnern verbinden.
2.2.2.1.1	Das HIL muss eine Verbindung zwischen ASAP und dem OBDH auf der Embedded-Hardware herstellen können.
2.2.2.1.2	Das HIL muss eine Verbindung zwischen ASAP und dem OBDH auf der PC-Hardware herstellen können.
2.2.2.1.3	Das HIL muss eine Verbindung zwischen ASAP und dem OBDH auf der internen Hardware im HIL herstellen können.
2.2.2.1.4	Das HIL muss eine Verbindung zwischen ASAP und Testanschlüssen zu Debug-Zwecken herstellen können.
2.2.2.1.5	Das HIL muss die Verbindungen auf Basis von Signalgruppen herstellen können.
2.2.2.2	Interne Logik-Plattform Das HIL muss eine interne Logik-Plattform bestehend aus Prozessor, Speicher, IOs und Bussen haben
2.2.2.3	Hardware-Schnittstellen (am Gehäuse)
2.2.2.3.1	Status LED für die Schaltverbindungen Das HIL soll seinen Schaltzustand mit Hilfe von LEDs auf einem Bedienerpanel anzeigen.
2.2.2.3.2	Abgriffe für das Oszilloskop Das HIL soll eine Schnittstelle für den Abgriff der Signale der aktiven Verbindung aufweisen.
2.2.2.3.3	Schnittstelle zu ASAP Das HIL muss eine Hardware-Schnittstelle zu ASAP aufweisen.
2.2.2.3.4	Schnittstelle zu OBDH auf der Embedded Hardware Das HIL muss eine Hardware-Schnittstelle zum OBDH auf der Embedded Hardware aufweisen.
2.2.2.3.5	Schnittstelle zu OBDH auf dem PC Das HIL muss eine Hardware-Schnittstelle zum OBDH auf dem PC aufweisen.
2.2.2.4	Software-Schnittstellen
2.2.2.4.1	Das HIL soll seinen Schaltzustand über eine Softwareanwendung einstellen lassen.

Tabelle 7-19: Systemanforderungen (funktionale Anforderungen) an das ATUS System

Die nachfolgende Tabelle 7-20 listet die Qualitätsanforderungen an das ATUS System.

ID	**Name und Begründung**
3	Systemanforderungen (Qualitätsanforderungen)
3.2	Anforderungen an das HIL
3.2.1	Anforderungen an das HIL seitens ASAP (als Testobjekt)
3.2.1.1	Schnittstellen
3.2.1.1.1	Energieversorgung Spannung: 6.00 V – 18.00 V

	Leistung: 6 W – 12 W
3.2.1.1.2	Digitale IO-Leitungen Anzahl: 32
3.2.1.1.3	UART-Bus Anzahl: 2
3.2.1.1.4	I2C-Bus Anzahl: 2
3.2.1.1.5	CAN-Bus Anzahl: 2
3.2.1.1.6	HS – Bus Anzahl: 4 (2x für OBC und 2x für PDH) => 16x LVDS-Signalpaarleitungen

Tabelle 7-20: Systemanforderungen (Qualitätsanforderungen) an das ATUS System

Die nachfolgende Tabelle 7-21 listet die Randbedingungen an das ATUS System

ID	Name und Begründung
4	Systemanforderungen (Randbedingungen)
4.2	Anforderungen an das HIL
4.2.1	Anforderungen an das HIL seitens des Lehrstuhls
4.2.1.1	Netzteil (Eingang) Das HIL darf die Leistungsgrenzen des Eingangsnetzteils nicht überschreiten. Spannung: 0V – 24V Leistung: max. 20W

Tabelle 7-21: Systemanforderungen (Randbedingungen) an das ATUS System

7.4.4 Systemkontext

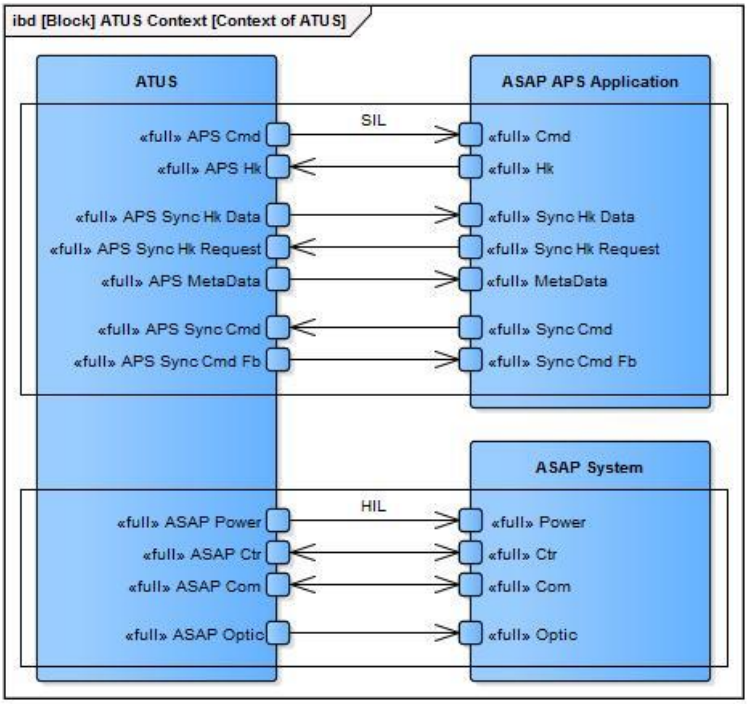


Abbildung 7-84: Systemkontext von ATUS

In den vorherigen Abschnitten wurden die Stakeholder identifiziert und die Anforderungen an das ATUS System analysiert. Auf der Grundlage dieser Vorarbeiten kann nun im nächsten Schritt der Kontext (siehe Abbildung 7-84) des zu entwickelnden Systems abgegrenzt werden. Durch die

Definition des Kontexts wird zum einen eine erste strukturierte Ordnung zwischen den beteiligten Systemen eingeführt, zum anderen wird jetzt klarer ersichtlich, welche Teilnehmer direkt miteinander interagieren und somit eine gemeinsame Schnittstelle aufweisen müssen.

ATUS ist eine Testumgebung. Aus diesem Grund sind die einzigen im Kontext enthaltenen Komponenten nur das Testsystem ATUS, bestehend aus den Simulationsumgebungen HIL und SIL, sowie die zwei Testobjekte ASAP und APS. Der Anschluss erfolgt über die Schnittstellen. Da die Testobjekte bereits in den Abschnitten 7.2 und 7.3 vorgestellt wurden, wird an dieser Stelle auf die Schnittstellen nicht weiter eingegangen.

7.4.5 Systemsynthese

Der nächste Schritt im Entwurfsprozess für das zu entwickelnde System ist die Synthese. Dabei soll, ausgehend vom Systemkontext und den an ihn gestellten Anforderungen, sein funktionales Verhalten realisiert werden. Hierbei müssen zwei Sichten berücksichtigt werden. Die erste bezieht sich auf das HIL (siehe Abbildung 7-85) mit ASAP als Testobjekt und die zweite auf das SIL (siehe Abbildung 7-86) mit APS als Testobjekt.

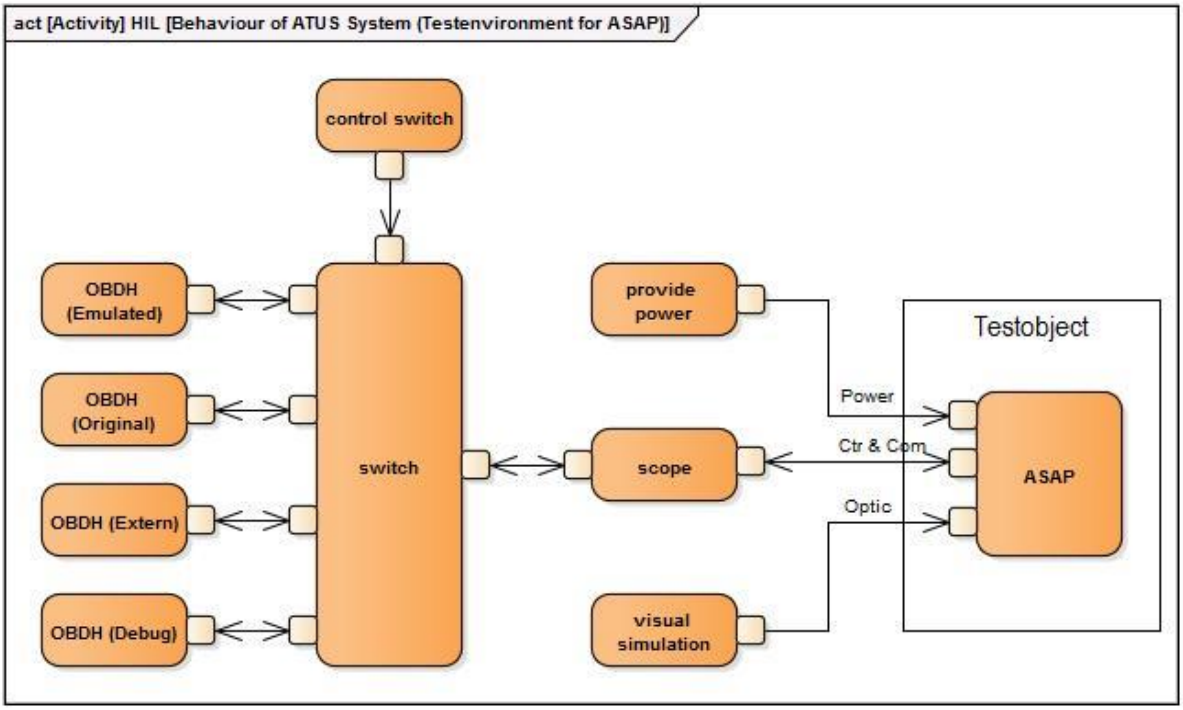


Abbildung 7-85: funktionales Verhalten des HILs von ATUS

Die Hauptaufgabe des HILs besteht darin, ASAP die Kommunikation mit seinen unmittelbaren Interaktionspartnern zu ermöglichen. Das wären der OBC (Onboard Computer) und das PDH (Payload Data Handling). Beide Einheiten werden zusammengefasst auch als *OBDH* (*Onboard Data Handling*) bezeichnet. Laut Anforderungen existieren mehrere Plattformen, die die OBDH-Funktionalität umsetzen. Der Tester soll daher die Möglichkeit haben, je nach Bedarf Verbindungen zwischen den einzelnen Plattformen und ASAP auf Basis von Signalgruppen (z.B. CAN1) herzustellen. Dazu ist ein Schaltelement (*switch*) nötig, das bidirektionale Signalverbindungen herstellen kann. Die Funktionsweise entspricht einem Multiplexer. Des Weiteren sollen die ein- und ausgehenden Signale von ASAP zu Analysezwecken (mit einem Oszilloskop) untersucht werden. Passende Abtaststellen (*scope*) können zwischen dem Schaltelement und ASAP angebracht werden. Um die Signale durch die Messung möglichst nicht zu belasten, sollten passende

Verstärkerschaltungen (Impedanzwandler) für den Abgriff verwendet werden. Ein passendes Netzteil sichert den Bedarf von ASAP hinsichtlich Leistung, Spannung und Strom. Die Steuerung (*control switch*) erfolgt hauptsächlich vom PC aus. Als letztes ist noch die visuelle Simulation (*visual simulation*) zu erwähnen. Das HIL muss für ASAP ein optisches Umfeld erzeugen, das durch dessen Kamera wahrgenommen wird.

Die bisherigen Betrachtungen bezogen sich auf physikalische Verschaltungen. Beim SIL (siehe Abbildung 7-86) hingegen werden Signalverbindungen auf Anwendungsebene (d.h. mittels RODOS-Topics und der RODOS-Middleware) hergestellt.

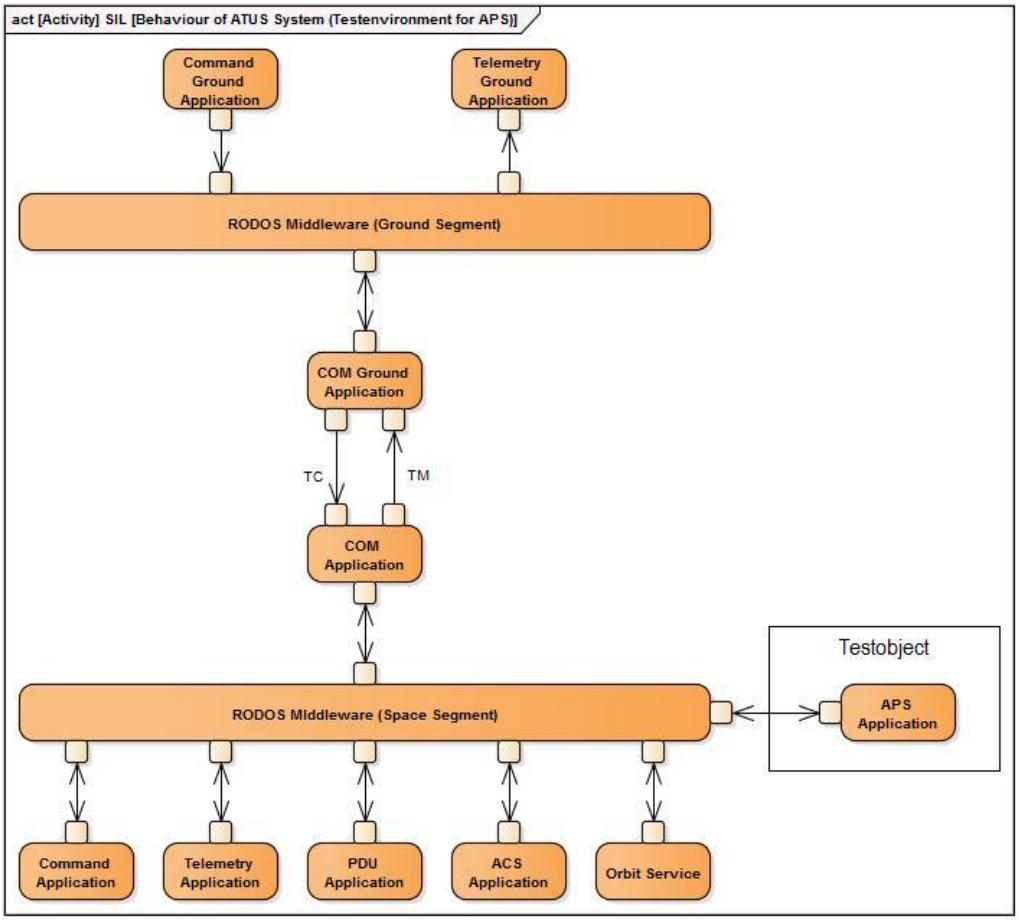


Abbildung 7-86: funktionales Verhalten des SILs von ATUS

Das Testobjekt ist in diesem Fall die APS Anwendung und läuft auf der Hardware (PC) des SIL-Simulators. Die anderen Anwendungen, die die Umgebung des APS darstellen, können entweder ebenfalls durch das SIL emuliert werden, oder auf einer der anderen durch das HIL geschalteten Plattform (z.B. OBDH – Original) portiert werden. Wegen der Abstraktion durch die Middleware ist diese Tatsache jedoch für das APS irrelevant. Das SIL erlaubt es, mehrere RODOS-Knoten in die simulierte Umgebung aufzunehmen. Um das Referenzszenario möglich getreu abzubilden, wurden zwei RODOS-Knoten, jeweils eines für das Bodensegment und das Raumsegment, vorgesehen. Die Kommunikation zwischen den beiden Segmenten erfolgt über die Kommunikationsanwendungen (*COM Application*, *COM Ground Application*) per Telekommandos (*TC*) und Telemetrie (*TM*). Die Umgebung des APS besteht größtenteils aus Softwareanwendungen, die die Subsysteme des Satelliten abbilden. Dazu gehören:

- Anwendung zur Kommandoverwaltung (Command Application) des Satelliten
- Anwendung zur Telemetrieverwaltung (Telemetry Application) des Satelliten
- Anwendung zur Energieversorgung (PDU Application) des Satelliten

- Anwendung zur Lageregelung (ACS Application) des Satelliten
- Dienst zur Berechnung der Positionen von Objekten auf einem bestimmten Orbit (Orbit Service)
- Anwendung zur Kommunikation (COM Application) des Satelliten
- Anwendung zur Kommunikation (COM Grond Application) am Boden
- Anwendung zur Kommandoverwaltung (Command Ground Application) am Boden
- Anwendung zur Telemetrieverwaltung (Telemetry Ground Application) am Boden

7.4.5.1 Hardware (HIL)

Das HIL, bzw. die Hardware der Testumgebung, besteht aus einem Gehäuse (siehe Abbildung 7-87) mit einem Front- und einem Backpanel.

Abbildung 7-87: Gehäuse des HIL (Hardware) – Ansicht auf das Frontpanel

Das Frontpanel dient primär als Schnittstelle zum Tester und erfüllt mehre Funktionen:

- Abgriffe der Signale (Bereich: links) zwischen ASAP und dem Schaltwerk
- Status Anzeige des Schaltwerks (Bereich: Mitte unten)
- Status Anzeige der digitalen IO-Signale von ASAP (Bereich: rechts)
- Status Anzeige und Schalter des HILs (Bereich: rechts oben)

Am Backpanel befinden sich die Anschlüsse, auf die der Tester keinen regelmäßigen Zugriff benötigt:

- Energieversorgung: 6V – 18V
- Anschluss von ASAP
- Anschluss des OBDH (Original)
- Anschluss des OBDH (Extern)
- Netzwerkanschluss zur Kommunikation mit der internen Logik
- 1x USB als Anschluss zum Programmieren und Debuggen der internen Logik
- 2x USB – Hostanschluss zur freien Verwendung der internen Logik

Im Inneren des Gehäuses befindet sich eine elektronische Platine (siehe Abbildung 7-88). Diese trägt den Hauptanteil der elektronischen Verschaltung, die aus folgenden Funktionsgruppen besteht:

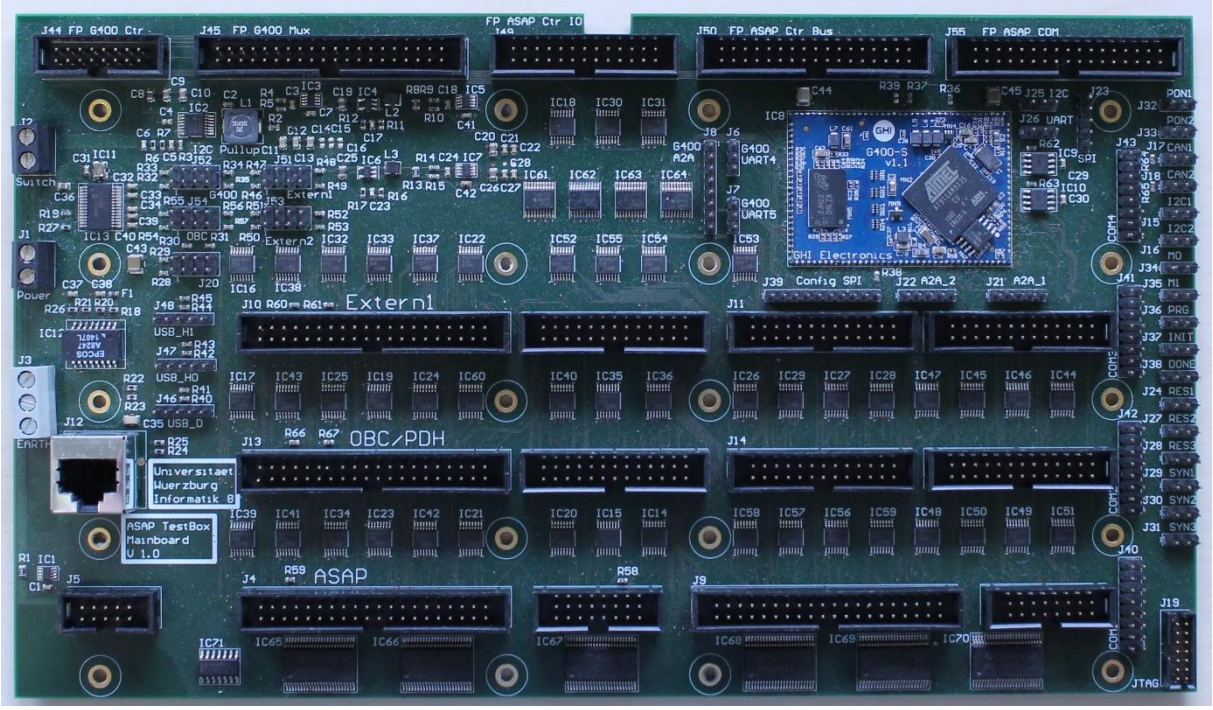


Abbildung 7-88: Platine des HIL (Hardware)

- Anschlüsse für die Komponenten des Front- und Backpanels.
- Eine interne Logik, die durch das G400-Modul von GHI realisiert wird.
- Ein Schaltwerk bestehend aus Multiplexern, die eine Verbindung zwischen ASAP und einem der vier zur Verfügung stehenden OBDH–Plattformen auf Basis von Signalgruppen herstellen. Es wird durch das G400-Modul gesteuert.
- Einen getakteten Spannungsregler, der eine stabile Ausgangsspannung von 3,3 V für die interne Logik und das Schaltwerk bereitstellt.
- Anschlüsse des OBDH (Debug).

Das G400 von GHI ist ein Prozessor-Modul (blauer Bereich der Platine), das mit externer Peripherie ausgestattet wurde. Dazu gehören unter anderem Speicher (92 MB RAM) und Kommunikationsbusse (z.B. CAN, I2C, SPI, Ethernet). Eine wichtige Besonderheit ist die Tatsache, dass das G400-Modul das .NET-Micro-Framework implementiert. Damit sich lassen sich auf dem Prozessor C#-Programme ausführen.

7.4.5.2 Software (SIL)

Das SIL, bzw. die Software der Testumgebung, ist ein dreischichtiges Programm, das für das Testobjekt die eigentliche Umgebung mitsamt den Anwendungen umsetzt. Es wurde in der Sprache C# (.NET - Framework 4.5) geschrieben und ist damit auf jedem Betriebssystem, das die .NET–Laufzeitsicht installiert hat, ausführbar. Zu den drei Schichten gehören eine Anwender-, eine Daten- und eine Persistenzschicht. Auf diese Schichten wird im folgenden Unterabschnitt näher eingegangen. Die Hauptaufgabe des SILs besteht darin, den Entwicklungsprozess von ASAP und APS ab einem möglichst frühen Stadium zu unterstützen. Dazu werden die dafür benötigten Anwendungen, insbesondere das APS als Testobjekt, als separate Komponenten in das C#-Programm eingebettet. Im weiteren Verlauf der Entwicklung können die Anwendungen schrittweise auf die eigentlichen Zielplattformen portiert werden.

Eine komplette und detaillierte Beschreibung des SILs würde den Rahmen der vorliegenden Dissertation überschreiten. Aus diesem Grund werden in den folgenden Unterabschnitten nur ausgewählte Aspekte behandelt.

7.4.5.2.1 Anwender-, Daten- und Persistenzschicht

Die graphische Benutzeroberfläche des SIL wurde mit Hilfe des Graphik-Frameworks WPF (Windows Presentation Foundation), das Bestandteil des .NET – Frameworks ist, erstellt. Sie entspricht der Anwenderschicht und ist in der folgenden Abbildung 7-89 dargestellt.

Abbildung 7-89: graphische Benutzeroberfläche

Im linken oberen Bereich befindet sich die Benutzerschnittstelle zur Hauptsteuerung des SILs. Damit kann der Anwender die Simulation starten, stoppen, pausieren, wiederaufnehmen und zurücksetzen. Des Weiteren kann dort auch der Zeitmodus (Echtzeit oder Zeitraffer) eingestellt werden. Unterhalb des Steuerungsbereichs ist der Arbeitsbereich (*Workspace*) angesiedelt. Dort sind alle Komponenten gelistet, die in der aktuellen Simulation berücksichtigt werden. Es gibt verschiedene Arten von Komponenten. Dazu gehören unter anderem: Anwendungen (z.B. APS oder Kommandoverwaltung), Dienste (z.B. Orbit-Dienst), Hardwareschnittstellen (z.B. COM/UART), RODOS-Topics. Die einzelnen Komponenten können per Drag&Drop in den mittleren Hauptbereich der graphischen Benutzeroberfläche gezogen werden. Dort werden sie im Detail dargestellt und können vom Anwender angepasst werden. Generell kommt bei der graphischen Benutzeroberfläche die Datenbindung der WPF zum Einsatz. Damit lassen sich die Daten der einzelnen Komponenten in der Datenschicht direkt auf der Oberfläche darstellen, editieren und laufend aktualisieren. Auf diese Weise kann der Anwender im laufenden Betrieb Einblick in den Zustand der jeweiligen Komponente erhalten. Diese Möglichkeiten müssen bei einer erfolgten Portierung der Komponente auf die Zielplattform nicht unbedingt gegeben sein und verdeutlichen daher die Stärken des SILs. Durch die Verwendung eines Reiters lassen sich auch gleichzeitig mehrere Komponenten im Hauptbereich verwalten. Neben der Datenbindung gibt es einen weiteren Mechanismus, mit dem Eingaben über die graphische Oberfläche zur Datenschicht weitergeleitet werden. Dabei handelt es sich um WPF-Kommandos, die der Datenschicht zugeordnet sind. Diese kapseln unter anderem Ereignisse (z.B. Bewegungen und Klicks mit der Maus) der graphischen Benutzeroberfläche mit Hilfe der Datenbindung. Dadurch gibt es

eine saubere Trennung zwischen der Präsentation (Anwenderschicht) und der verarbeitenden Logik (Datenschicht).

Die Datenschicht ist streng hierarchisch aufgebaut. An oberster Stelle steht das Objekt, das den gesamten Arbeitsbereich (*ATUS Workspace*) kapselt. Alle anderen Objekte der Datenschicht sind ihm entweder direkt oder indirekt untergeordnet. Durch diese strukturelle Ordnung können mittels Traversierung sämtliche Aktionen des Simulators (z.B. start, stop) vom Arbeitsbereich aus an alle untergeordneten Komponenten weitergereicht werden.

Für die nichtflüchtige Speicherung der Daten ist die Persistenzschicht verantwortlich. Insgesamt sind zwei Speicherorte vorgesehen: Die erste Möglichkeit besteht darin, dass der Anwender die Daten, d.h. den gesamten Arbeitsbereich, auf den lokalen Datenträger als Datei hinterlegt und zu einem späteren Zeitpunkt wieder lädt. Dieser Vorgang setzt eine Serialisierung bzw. Deserialisierung der Daten voraus. Der Simulator verwendet ein binäres Verfahren, das vom .NET-Framework zur Verfügung gestellt wird. Die andere Möglichkeit besteht darin, die Daten auf einem Datenbankserver zu sichern. Hierfür müssen SQL-Abfragen, in Abhängigkeit der Datenschichtstruktur, erstellt und verwaltet werden. Mit der Verwendung des Entity-Frameworks 6.0 wird diese Aufgabe zum Teil automatisiert.

7.4.5.2.2 Umgebungssimulation und 3D – Visualisierung

Das SIL bietet dem Anwender die Möglichkeit, verschiedene Referenzszenarien zu simulieren und optisch darzustellen (siehe Abbildung 7-90). Die dafür notwendigen Funktionalitäten sind in einer eigenen Raumsimulation (*ATUS SpaceSim)* ausgelagert.

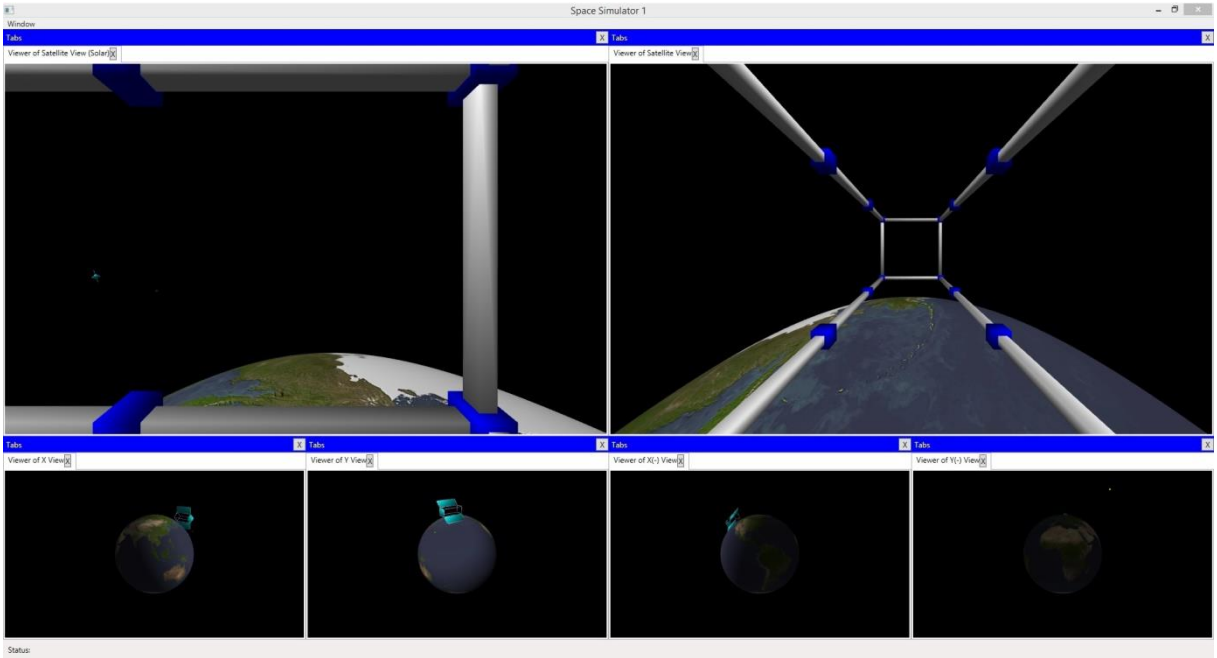


Abbildung 7-90: 3D - Visualisierung

Eine Besonderheit der Visualisierung ist, dass sich für die 3D-Szene der Raumsimulation verschiedene Ansichten gleichzeitig berechnen lassen. Eine Ansicht entspricht dem Blickfeld einer Kamera. Damit kann das Szenario / die Szene gleichzeitig aus unterschiedlichen Blickwinkeln betrachtet werden. Des Weiteren lassen sich in die Szene verschiedene Objekte (Satelliten, Raumschrott, Erde, Sonne, Mond, Koordinatenachsen) einfügen. Ihre Bewegung kann bestimmten Bahnen folgen. Die räumliche Ausrichtung ist ebenfalls einstellbar.

Die Raumsimulation besitzt eine statische Struktur, die in der folgenden Abbildung 7-91 dargestellt ist.

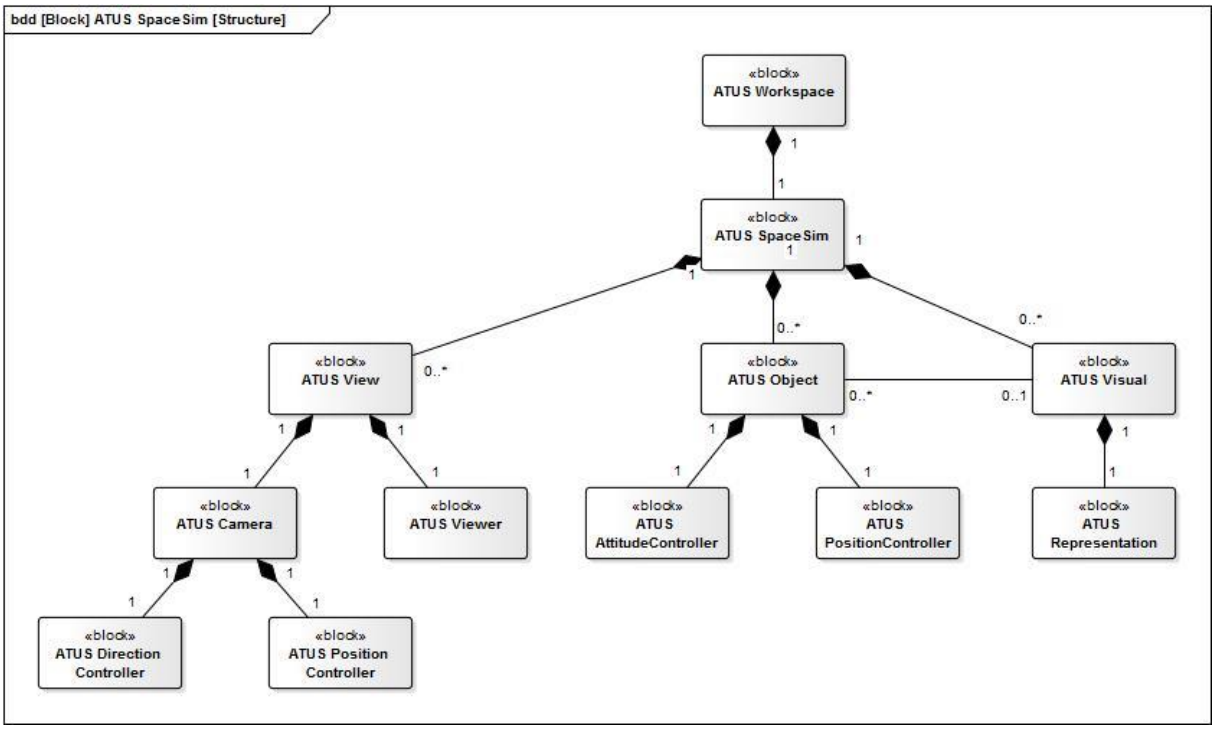


Abbildung 7-91: statische Struktur der Raumsimulation

Die Raumsimulation (*ATUS SpaceSim*) ist ein direkter Bestandteil des SILs bzw. des Arbeitsbereichs (*ATUS Workspace*). Sie kann, wie bereits erwähnt wurde, über mehrere Ansichten (*ATUS View*) verfügen. Des Weiteren werden in ihr Objekte (*ATUS Object*) und ihre graphischen Darstellungen (*ATUS Visual*) verwaltet. Ein Objekt kapselt physikalische Informationen, wie zum Beispiel Position und Ausrichtung.

Für jede dieser Ansichten muss eine Kamera (*ATUS Camera*) erstellt werden. Zu den wichtigsten Einstellungen der Kamera gehören die Sichtweite (Field Of View), die Position (*ATUS Position Controller*) und die Sichtrichtung (*ATUS Direction Controller*). Die Angaben zur Position können sich auf verschiedene Koordinatensysteme beziehen. Für jede Variante gibt es einen abgeleiteten Positionskontroller:

- Koordinaten im ECI (Earth Centered Inertial)
  Die Position der Kamera bleibt während der Simulation an einem Ort fixiert.
- Koordinaten im ECF (Earth Centered Fixed)
  Die Position der Kamera (bezogen auf das Inertialsystem der Erde) wandert mit der Drehung der Erde
- Koordinaten im Bezugssystem zu einem Objekt (*ATUS Object*)
  Die Position der Kamera (bezogen auf das Inertialsystem der Erde) wandert mit der Bewegung des Objektes auf seiner Bahn

Die Sichtrichtung einer Kamera lässt sich ebenfalls auf verschiedene Weise einstellen:

- Sichtrichtung anhand eines Vektors im ECI
  Die Sichtrichtung der Kamera bleibt konstant
- Sichtrichtung anhand eines Vektors im ECF
  Die Sichtrichtung der Kamera (bezogen auf das Inertialsystem der Erde) wandert mit der Drehung der Erde

- Nachgeführte Sichtrichtung anhand eines Zielpunktes im ECI oder ECF
  Die Sichtrichtung der Kamera wird so nachgeführt, dass der Zielpunkt sich stets im Mittelpunkt des Sichtfeldes befindet
- Nachgeführte Sichtrichtung anhand einer konstanten Richtungsvorgabe bezogen auf das Bezugssystem zu einem Objekt
  Das Sichtfeld wird an die Ausrichtung eines Objektes fixiert. Wenn sich das Objekt dreht, dann dreht sich auch die Kamera in gleichem Maße.

Des Weiteren muss jede Ansicht über eine graphische Darstellung (*ATUS Viewer*) verfügen. In dieser sind die Funktionalitäten gekapselt, die die Szene visuell aufbauen. Es gibt mehrere Möglichkeiten, dies zu realisieren. Daher existieren mehrere Ableitungen von *ATUS Viewer*. Jede konkrete Variante greift in der Regel auf eine Graphik-Bibliothek zurück:

- Viewport3D-Steuerelement der WPF
  Verwendet intern DirectX. Diese Art der Darstellung wird standardmäßig verwendet.
- Native DirectX-Schnittstelle
  Wurde bisher noch nicht implementiert.
- Native OpenGL-Schnittstelle
  Wurde bisher noch nicht implementiert.
- Externer Graphik-Simulator
  Die 3D-Szene wird mit Hilfe eines externen Graphik-Simulators (PC) dargestellt. Die notwendigen Daten zum Aufbau der Szene werden zuvor per Netzwerkverbindung übertragen.

Eine Szene beinhaltet Objekte (*ATUS Objekt*), die dargestellt werden sollen. Jedes einzelne Objekt verfügt über zwei Kontroller, mit denen jeweils die Position (*ATUS Position Controller*) und die räumliche Ausrichtung (*ATUS Attitude Controller*) eingestellt wird. Analog zu der Kamera existieren auch hier verschiedene Umsetzungsmöglichkeiten:

- Koordinaten in ECI (Earth Centered Inertial)
  Die Position des Objektes bleibt während der Simulation an einem Ort fixiert.
- Koordinaten durch Bahnmodelle (z.B. Kepler, SGP4)
  Die Position des Objektes wird durch Bahnparameter berechnet

Die räumliche Ausrichtung eines Objekts lässt sich ebenfalls auf verschiedene Weise einstellen:

- Konstante Ausrichtung anhand eines Vektors in ECI
  Die räumliche Ausrichtung bleibt konstant.
- Nadir
  Das Objekt wird so ausgerichtet, dass ein definierter Vektor im Bezugssystem des Objekts auf den Mittelpunkt der Erde zeigt.
- Point
  Das Objekt wird so ausgerichtet, dass ein definierter Vektor im Bezugssystem des Objekts auf einen definierten Punkt (in ECI oder ECF) im Raum zeigt.
- Extern
  Das Objekt wird von einer Komponente außerhalb des Raumsimulators (z.B. das Lage-regelungssystem eines Satelliten) gesteuert.

Die Algorithmen zur Berechnung der Ausrichtung einer Kamera oder eines Objektes entsprechen denen der Lageregelung (siehe Abschnitt 7.4.5.2.3).

7.4.5.2.3 Lageregelung

Zur Realisierung der Referenzszenarien (siehe Abschnitt 7.1) wird eine aktive Lageregelung benötigt. Diese Aufgabe fällt in den Verantwortungsbereich des Lageregelungssystems (ACS: Attitude Control System) des Satelliten. Im SIL wird dieses Subsystem als eine Softwareanwendung abgebildet, die direkt die Lage des Satelliten-Objektes in der Umgebungssimulation (siehe Abschnitt 7.4.5.2.2) beeinflussen kann.

Die Lageregelung muss im Betrieb mehrere Funktionsweisen aufweisen. Generell geht es darum, ein Quell-Koordinatensystem, das die räumliche Lage und Ausrichtung eines Quell-Objektes (z.B. Kamera, Antenne) im Satelliten angibt, auf einen Zielpunkt im Raum (z.B. ein Phänomen in der Erdatmosphäre, Bodenstation) auszurichten. Insgesamt sind dabei vier verschiedene Koordinatensysteme zu berücksichtigen:

- ECI-Frame
  Das ECI-Frame (Earth Centered Inertial) hat seinen Ursprung im Erdmittelpunkt. Die X-Achse ergibt sich aus dem Schnitt der Ekliptik- und der Äquatorialebene und zeigt in Richtung Frühlingspunkt. Die Z-Achse zeigt in Richtung Nordpol. Die Y-Achse ergibt sich aus dem Kreuzprodukt von X- und Z-Achse. Das Koordinatensystem wird als ruhend betrachtet.
- ECF-Frame
  Das ECF-Frame (Earth Centered Fixed) hat seinen Ursprung im Erdmittelpunkt. Die X-Achse liegt auf der Äquatorialebene und zeigt in Richtung Nullmeridian (Greenwich). Die Z-Achse zeigt in Richtung Nordpol. Die Y-Achse ergibt sich aus dem Kreuzprodukt von X- und Z-Achse. Das Koordinatensystem wandert in Bezug auf das ECI mit der Drehung der Erde. Der Winkel zwischen den X-Achsen beider Koordinatensysteme kann von der Sternzeit abgeleitet werden.
- SAT-Frame
  Das SAT-Frame (Satellite) bezieht sich auf den Satelliten. Dessen Massenzentrum dient als Ursprung des Koordinatensystems. X-, Y- und Z-Achsen sind zueinander orthogonal, können aber ansonsten willkürlich festgelegt werden. Das Koordinatensystem wandert in Bezug auf das ECI mit der Bahnbewegung des Satelliten und ändert seine Orientierung mit der Drehbewegung des Satelliten.
- SRC-Frame
  Das SRC-Frame (Source) bezieht sich auf das eigentliche im Satelliten befindliche Objekt, das durch die Lageregelung indirekt (d.h. mit dem Satelliten) ausgerichtet werden soll. Der Ursprung und die Achsen können willkürlich festgelegt werden. Letztere müssen jedoch zueinander orthogonal sein. Das Koordinatensystem ist gegenüber dem SAT fixiert und wandert in Bezug auf das ECI mit der Bahnbewegung des Satelliten.

Die nachfolgende Abbildung 7-92 zeigt die Transformationsmöglichkeiten zwischen den Koordinatensystemen. Eine Transformation besteht aus einer Rotation und einer anschließenden Translation. Alle dafür nötigen Parameter (z.B. Lage des Quellobjekts im Satelliten) sind gegeben.

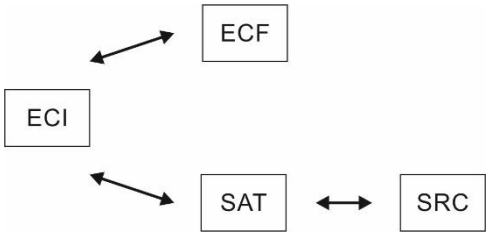


Abbildung 7-92: Transformationen zwischen Koordinatensystemen

Bedingt durch die Referenzszenarien muss die Lageregelung mehrere unterschiedliche Betriebsmodi (*Mode*) unterstützen:

- Horizon-Mode
  Der Horizont-Modus dient dazu, das SRC-Frame eines Quell-Objekts auf einen Ziel-Punkt über den Horizont auszurichten. Die zwei folgenden Abbildungen (Abbildung 7-93 und Abbildung 7-94) veranschaulichen ein Beispiel mit einer Kamera als Quellobjekt. Die Achse, die auf das Ziel ausgerichtet ist, entspricht in diesem Fall der Sichtrichtung (FOV, Field of View) der Kamera. Des Weiteren wird das SRC-Frame so ausgerichtet, dass sich der Horizont im Bild zentral im unteren Bereich befindet (siehe Orientierung im Absatz unten). Der Ziel-Punkt kann durch den Modus so berechnet werden, dass die Maße e und h in einem gewünschten Verhältnis zueinander stehen.

$$0 \leq \frac{e}{h} \leq 1$$

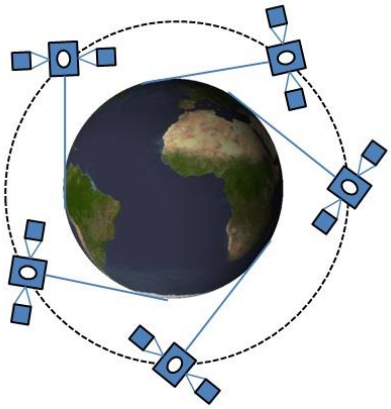


Abbildung 7-93: Horizont-Modus (Außen-Ansicht)

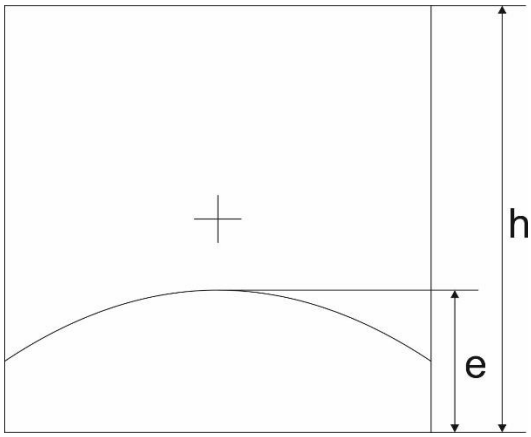


Abbildung 7-94: Horizont-Modus (FOV-Ansicht)

- Pointing-Mode
  Der Ziel-Modus dient dazu, das SRC-Frame eines Quell-Objekts auf einen beliebigen Ziel-Punkt auszurichten. Die Zielkoordinaten können sich dabei auf ein beliebiges Referenzsystem (z.B. Bodenstation in ECF oder Sonne im ECI) beziehen.
- Sun-Mode
  Der Sonnen-Modus verhält sich wie der Ziel-Modus. Das Ziel ist die Sonne. Das SRC-Frame hängt vom Einbau der Solarpanele auf dem Satelliten ab.
- Communication-Mode
  Der Kommunikations-Modus verhält sich ebenfalls wie der Ziel-Modus. Das Ziel ist die Bodenstation auf der Erde. Das SRC-Frame hängt vom Einbau der Antennen im Satelliten ab.

Die Lageregelung verwendet für alle Modi denselben Algorithmus, da die Aufgabe immer darin besteht, das Quell-Objekt auf ein Ziel-Objekt auszurichten. Folgende zwei Bedingungen müssen für eine Ausrichtung erfüllt sein:

- Die X-Achse des SRC-Frames muss auf das Zielobjekt zeigen.
- Die Y-Achse des SRC-Frames muss in der Ebene liegen, die durch die X-Achse des SRC-Frames und durch einen vorgegebenen Orientierungsvektor (R-Vektor) aufgespannt wird.

Der Algorithmus verfügt über einige Parameter, die durch die verschiedenen Modi gesetzt werden müssen:

- Die Translation und Rotation des SRC-Frames gegenüber dem SAT-Frame

- Die Translation und Rotation des SAT-Frames gegenüber dem ECI-Frame
- Die Zielkoordinaten
- Die Orientierung (R-Vektor)

Da angenommen wird, dass im Satelliten das Quell-Objekt starr verbaut wurde, besteht die einzige Möglichkeit, dieses auf das Ziel auszurichten, darin, dass der Satellit seine Lage, nicht jedoch seine Position, mit Hilfe seiner Lageaktuatoren (z.B. Reaktionsrad) ändert. Die nachfolgenden zwei Abbildungen (Abbildung 7-95 und Abbildung 7-96) verdeutlichen die Thematik.

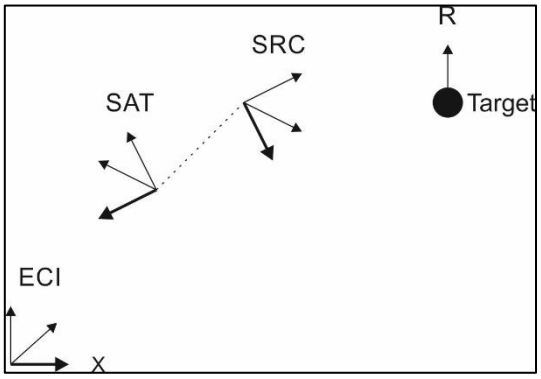


Abbildung 7-95: vor der Ausrichtung

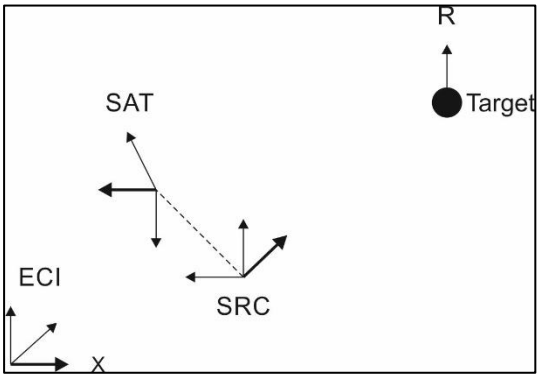


Abbildung 7-96: nach der Ausrichtung

Der Algorithmus muss aus folgenden Gründen kontinuierlich Stellwerte für die Aktuatoren berechnen und dadurch eine Nachführung ermöglichen:

- Das Zielobjekt bewegt sich auf einer Bahn und ändert daher laufend seine Koordinaten.
- Der Satellit mit dem Quell-Objekt bewegt sich ebenfalls auf einer Bahn und ändert daher laufend seine Koordinaten.
- Der Satellit mit dem Quell-Objekt besitzt einen Drehimpuls und ändert daher laufend seine Lage.

Das SRC-Frame wird durch folgende Matrix definiert:

$$SRC_{SAT} = \begin{pmatrix} DirX_X & DirY_X & DirZ_X & Off_X \\ DirX_Y & DirY_Y & DirZ_Y & Off_Y \\ DirX_Z & DirY_Z & DirZ_Z & Off_Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Spaltvektoren *DirX, DirY, DirZ* sind orthonormal. Die Matrix beschreibt eine Rotation mit anschließender Translation (*Off*) des SRC-Frames (Quell-Objekts) gegenüber dem SAT-Frame (Satelliten).

Aus diesen Angaben lassen sich drei weitere Winkel herleiten:

$$\alpha = \sphericalangle(DirX, Off)$$

$$\beta = \sphericalangle(DirY, Off)$$

$$\gamma = \sphericalangle(DirZ, Off)$$

Zur Berechnung der Ausrichtung müsste der Algorithmus folgendes nicht-lineares Gleichungssystem mit 12 Unbekannten lösen.

1. $DirNewX_X^2 + DirNewX_Y^2 + DirNewX_Z^2 - 1 = 0$
2. $DirNewY_X^2 + DirNewY_Y^2 + DirNewY_Z^2 - 1 = 0$
3. $DirNewZ_X^2 + DirNewZ_Y^2 + DirNewZ_Z^2 - 1 = 0$

4. $\overrightarrow{DirNewX} \cdot \overrightarrow{DirNewY} = 0$
5. $\overrightarrow{DirNewX} \cdot \overrightarrow{DirNewZ} = 0$
6. $\overrightarrow{DirNewY} \cdot \overrightarrow{DirNewZ} = 0$
7. $OffNew_X^2 + OffNew_Y^2 + OffNew_Z^2 - |\overrightarrow{Off}|^2 = 0$
8. $\overrightarrow{DirNewY} \cdot (\overrightarrow{Target} - \overrightarrow{OffNew}) = 0$
9. $\overrightarrow{DirNewZ} \cdot (\overrightarrow{Target} - \overrightarrow{OffNew}) = 0$
10. $\overrightarrow{DirNewY} \cdot \overrightarrow{DirNewX}\ x\ \vec{R} = 0$
11. $\overrightarrow{-OffNew} \cdot \overrightarrow{DirNewX} - \cos(\alpha) \cdot |\overrightarrow{-OffNew}| = 0$
12. $\overrightarrow{-OffNew} \cdot \overrightarrow{DirNewY} - \cos(\beta) \cdot |\overrightarrow{-OffNew}| = 0$

Das Ziel besteht darin, ein neues Referenzsystem (Frame) für das Quell-Objekt mit den Größen *DirXNew, DirYNew, DirZNew* und *OffNew* zu bestimmen, das die oben genannten Bedingungen der Ausrichtung erfüllt. Die ersten drei Gleichungen legen fest, dass es sich bei den Vektoren *DirNewX, DirNewY, DirNewZ* um Einheitsvektoren handelt. Die nächsten Gleichungen mit den Nummern 4 bis 6 definieren, dass die zuvor erwähnten Vektoren zueinander orthogonal sind. Gleichung 7 definiert, dass der Ursprung des neuen zu bestimmenden Referenzsystems vom SAT-Frame genau denselben Abstand wie das ursprüngliche Referenzsystem aufweisen muss. Die nachfolgenden Gleichungen mit den Nummern 8 und 9 legen fest, dass die X-Achse des neuen Referenzsystems genau auf das Ziel ausgerichtet ist. Gleichung 10 richtet den UP-Vektor (Y-Achse) so aus, dass dieser Teil der Ebene ist, die durch den R-Vektor und der neuen X-Achse aufgespannt wird (Orientierung). Die letzten zwei Gleichungen (11 und 12) berücksichtigen, dass das Quell-Objekt im Satelliten auf eine bestimmte Weise (Winkel) verbaut wurde.

Zusammengefasst lässt sich das neue Referenzsystem in folgender Form darstellen:

$$SRC_New_{SAT} = \begin{pmatrix} DirNewX_X & DirNewY_X & DirNewZ_X & OffNew_X \\ DirNewX_Y & DirNewY_Y & DirNewZ_Y & OffNew_Y \\ DirNewX_Z & DirNewY_Z & DirNewZ_Z & OffNew_Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In einem letzten Schritt kann nun die Rotations-Matrix für den Satelliten hergeleitet werden. Aus ihr lässt sich entnehmen, wie der Satellit sich drehen muss, damit das ursprüngliche Referenzsystem für das Gerät in das neue Referenzsystem überführt wird. Dazu ist folgende mathematische Operation nötig:

$$SRC_New_{SAT} = \begin{pmatrix} 0 & 0 & 0 & OffNew_X \\ 0 & 0 & 0 & OffNew_Y \\ 0 & 0 & 0 & OffNew_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot Rot_Sat \cdot \begin{pmatrix} 0 & 0 & 0 & -Off_X \\ 0 & 0 & 0 & -Off_Y \\ 0 & 0 & 0 & -Off_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot SRC_{SAT}$$

Um das neue Referenzsystem zu erhalten, muss das alte in den Ursprung des Satelliten verschoben werden. Anschließend erfolgt die zu bestimmende Rotation des Satelliten. Zuletzt muss das Referenzsystem auf seinen neuen Offset gesetzt werden. Durch Equivalenzumformungen der letzten Gleichung kann explizit die Rotationsmatrix für den Satelliten bestimmt werden:

$$Rot_Sat = \begin{pmatrix} 0 & 0 & 0 & OffNew_X \\ 0 & 0 & 0 & OffNew_Y \\ 0 & 0 & 0 & OffNew_Z \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \cdot SRC_New_{SAT} \cdot SRC_{SAT} \cdot \begin{pmatrix} 0 & 0 & 0 & -Off_X \\ 0 & 0 & 0 & -Off_Y \\ 0 & 0 & 0 & -Off_Z \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}.$$

7.5 Modellbasierte Systementwicklung

Wie im Kapitel 5 (Methode) erläutert wurde, baut der angewendete Entwurfsprozess auf der modellbasierten Systementwicklung (engl. MBSE, model-based-system-engineering) auf. Der Grundgedanke dieser Vorgehensweise besteht darin, dass alle relevanten Daten (Quellcode, Dokumentation, usw.) von einem zentralen Modell abgeleitet werden. Änderungen am Design sind grundsätzlich nur am Modell vorzunehmen. Der Vorteil dieser Entwurfsmethode besteht darin, dass alle (abgeleiteten) Daten zueinander konsistent bleiben (z.B. Quellcode zur Dokumentation). Die nachfolgende Abbildung 7-97 zeigt eine Übersicht, wie die MBSE im Rahmen der Vorhaben ASAP (inkl. APS) und ATUS umgesetzt wurden.

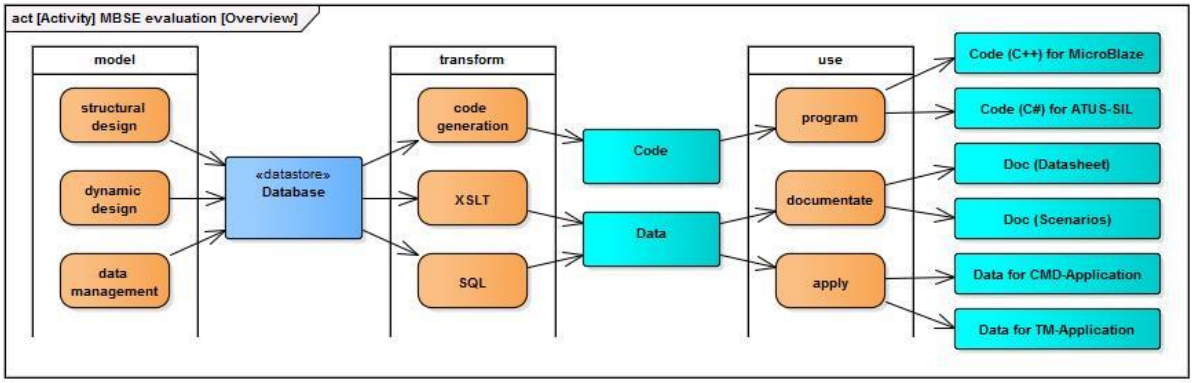


Abbildung 7-97: Übersicht der modellbasierten Systementwicklung

Die MBSE konnte nicht vollständig auf alle Bereiche der Systementwicklung angewendet werden, da sich insbesondere der Datenfluss der Anwendungen zur Entwicklung der Hardware nur mit erheblichem Aufwand in das Modell integrieren ließe. Die entsprechend verwendeten Programme sind hochspezialisiert und schwer anzupassen. Daher musste die MBSE mit einem klassischen Ansatz kombiniert werden. Dennoch betraf diese Einbuße lediglich die Implementierungsphase (z.B. Erstellung der elektronischen Schaltpläne). In der Designphase konnte der MBSE-Ansatz (auch für die Hardware) verfolgt werden. Die abgeleiteten Daten können z.B. in Form der in den letzten Abschnitten dargestellten SysML-Diagrammen eingesehen werden. Auf der Softwareseite konnte die MBSE zwar konsequenter als bei der Hardware über alle Entwicklungsphasen angewendet werden. Jedoch war es nicht möglich, sämtlichen Output, insbesondere den Quellcode für die Steuerflüsse, durchgehend abzuleiten. In den folgenden Unterabschnitten werden die entsprechenden Hindernisse aufgezeigt.

Grundsätzlich sollten mit der MBSE verschiedene Arten von Output generiert werden können:

- Quellcode (C++, C#) für die verschiedenen Plattformen (Xilinx MicroBlaze, PC) zur Programmierung
- Dokumentation (Datenblatt, Szenario- und Konfigurationsbeschreibungen) für verschiedene Zielgruppen

Um die oben genannten Arten von Output aus dem Modell generieren zu können, müssen verschiedene Transformationen eingesetzt werden. Generell müssen dabei entweder Code oder Datenpakete abgeleitet werden. Dazu kommen verschiedene Technologien (siehe folgende Unterabschnitte) zum Einsatz:

- Automatische Quellcode-Generierung
- XSLT: Extensible Stylesheet Language Transformation
- SQL: Structured Query Language

Das Modell selbst muss die Struktur, das Verhalten und mögliche Konfigurationen (z.B. Einstellungen bzgl. des autonomen Planungsverhaltens) des Systems abbilden und nicht-flüchtig speichern können.

Dazu kommt, wie im Kapitel 5 (Methode) bereits erläutert wurde, das Modellierungsprogramm „Enterprise Architekt“ zum Einsatz. Diese Modellkonstruktion wird zusätzlich mit einer relationalen Datenbank, die auf einem MS SQL Server läuft, ergänzt.

Zum besseren Verständnis werden in den folgenden Unterabschnitten ausgewählte Prozesspfade vom Modell bis zum jeweiligen Output im Detail beschrieben.

7.5.1 Modellierung des Systems

Wie in Abbildung 7-97 dargestellt wurde, besteht die Modellierung eines Systems (z.B. ASAP) im Wesentlichen aus drei verschiedenen Vorgängen:

- Design der Systemstruktur
  (Beispiel: eine Anwendung – z.B. ASAP – produziert 0 … n Telemetriewerte)
- Design des Systemverhaltens
  (Beispiel: die Telemetriewerte werden zwischengespeichert, gruppiert und dann gesendet)
- Datenverwaltung
  (Erzeugung und Persistierung der durch die Systemstruktur definierten Instanzen)

Die ersten beiden Vorgänge wurden mit Hilfe des Modellierungstools „Enterprise Architekt“ von „Sparx Systems“ unter der Verwendung der Modellierungssprache SysML (siehe Abschnitt 2.4) realisiert. Mit ihr lassen sich verschiedene Typen von Diagrammen (z.B. Blockdiagramme) erstellen, die im weiteren MBSE-Entwurfsprozess als Output ausgegeben werden. Im Kapitel 7 (Realisierung) kommen fast ausschließlich solche Diagramme zum Einsatz. Dennoch reicht die Definition der Systemstruktur allein nicht aus. Sie beschreibt lediglich, wie die verschiedenen Typen von Daten zueinander in Relation stehen. Die Modellierung muss um eine Datenverwaltung ergänzt werden. Mit dieser lassen sich die einzelnen in der Struktur definierten Daten instanzieren und persistent abspeichern. Zu diesem Zweck eignet sich besonders eine relationale Datenbank. Konkret kam der „MS SQL Server 2016 Express“ zum Einsatz. Getreu der MBSE wird in „Enterprise Architekt“ aus der bereits definierten Systemstruktur ein relationales Datenbankmodell mit seinen beschreibenden Tabellen, Relationen und Attributen abgeleitet. Dieses Datenbankmodell wird dann in einem weiteren Schritt mit einer real existierenden Datenbank auf dem Server synchronisiert. Im Hintergrund werden dabei automatisch SQL-Befehle der DDL (Data Definition Language) erstellt und ausgeführt. Die Manipulation der eigentlichen Daten erfolgt komfortabel über eine GUI. Die Bindung zwischen der Datenbank und der GUI wird mit Hilfe des Entity Frameworks von Microsoft realisiert. Diese leitet aus dem Schema der Datenbank eine passende Datenschicht ab, die als Vermittler zwischen Datenbank und GUI dient. Die nachfolgende Abbildung 7-98 zeigt die genauen Zusammenhänge.

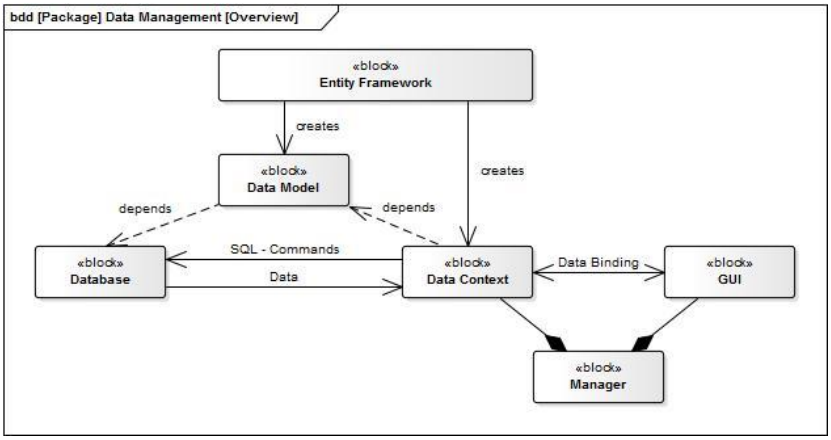


Abbildung 7-98: Datenverwaltung

Im vorliegenden Fall wird der sogenannte „Database First“ – Ansatz des Entity-Frameworks angewendet. Ausgehend aus einer bereits vorliegenden Datenbank leitet das Framework ein Datenmodell ab. Dieses besteht aus drei XML-Dateien: eine, um die Struktur (im Framework auch Schema genannt) der Datenbank zu beschreiben; eine, um die Struktur des Datenmodells abzubilden; eine, um die Beziehungen (im Framework auch Mapping genannt) zwischen den beiden Schemata herzustellen. Das so erzeugte Datenmodell bildet die Grundlage, um in einem weiteren Schritt mit Hilfe sogenannter T4-Transformationen eine passende Hierarchie an Klassen abzuleiten, die für die anschließende Datenbindung zur GUI geeignet sind. An oberster Stelle der Hierarchie steht der sogenannte Datenkontext (DataContext). In diesem sind die einzelnen Tabellen der Datenbank als Datenmengen (DbSet) abgebildet. Zu Beginn der Anwendung werden die Daten der Datenbank vom Kontext geladen und lokal zwischen gespeichert. Die GUI, die den Kontext mittels einer Datenbindung abruft und visualisiert, verbindet sich daher immer nur mit einer lokalen Kopie der Daten. Eingaben des Anwenders werden von der GUI ebenfalls entgegen genommen, mittels Datenbindung im Kontext zwischengespeichert, und dann (ebenfalls erst auf Wunsch des Anwenders) anschließend zur Datenbank übertragen. Eine Anwendung, auch Manager genannt, kapselt den Datenkontext und die GUI, und vereinigt somit Daten- und Präsentationsschicht. Außerdem ist in ihr die Logik zur weiteren Verarbeitung (siehe nachfolgend Abschnitte 7.5.2 und 7.5.3) implementiert.

Die Nutzung des Entity-Frameworks bietet im Wesentlichen folgende Vorteile:

- Die T4 – Transformation entspricht in ihrem Wirken der MBSE
- Die T4 – Transformation ist einheitlich und wird für alle Tabellen der Datenbank (bzw. DbSets des Kontexts) angewendet. Eine Erweiterung der Datenbank (bzw. des Datenmodells) ist daher mit sehr geringem Aufwand verbunden. Es muss kein Quellcode manuell erstellt werden.
- Mittels der Abstrahierung über ein Datenmodell ist die Datenbank für die GUI transparent und kann bei Bedarf gegen eine andere Datenquelle (nicht nur relationale Datenbanken) ausgetauscht werden. Diese Option erlaubt einen Wechsel auf die Datenbank-Infrastruktur eines anderen Projektes/Vorhabens (z.B. SONATE)

Für die Vorhaben ASAP, ATUS und SONATE wurde eine Datenbank, die logisch in zwei Schemata untergliedert ist, erstellt. Das erste Schema mit dem Namen „Top“ (siehe Abbildung 7-99) beschränkt sich auf die Beziehungen der einzelnen Komponenten von ASAP und dem Satelliten (SONATE). Das zweite Schema mit dem Namen „APS“ bezieht sich ausschließlich auf das APS. Es besteht daher überwiegend aus den in den Abschnitten des Kapitels 7 (Realisierung) vorgestellten Komponenten. Da deren Beziehungen bereits bekannt sind, wird das zugehörige Schema nicht erneut extra dargestellt.

Die nachfolgende Auflistung bietet eine grobe Gliederung der zwei Schemata:

- APS
  - Datenverwaltung (z.B. Nachrichten und Signale)
  - Ereignisverwaltung (z.B. Nachrichten und Ereignisse)
  - Entscheidungsfindung (z.B. Entscheidungen, Trigger)
  - Ablaufplanung (z.B. Definition von Aktivitäten und Aktionen)
  - Planausführung (z.B. Kommandos und Parameter)

- ASAP u. SONATE
  - Satellit, Subsystem, Anwendungen, Module
  - Datenwerte (z.B. Zustände einzelner Module, Werte von Sensoren)
  - Datenlisten (z.B. Konfigurationseinstellungen des APS)
  - Kommandos
  - Housekeeping (Standard Online/Offline)
  - Housekeeping (Extended)

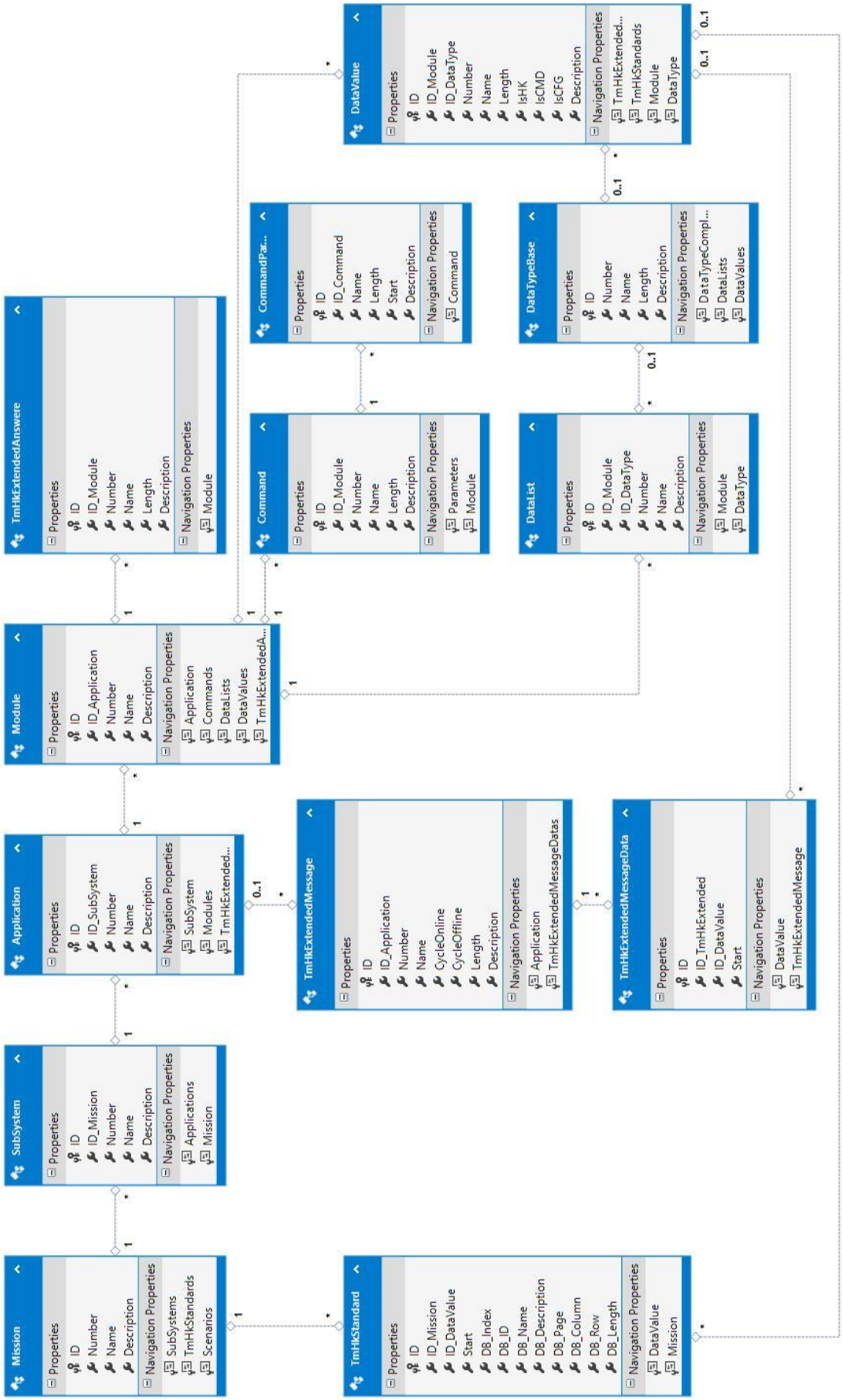


Abbildung 7-99: Datenmodell für ASAP und SONATE (Schema Top)

7.5.2 Automatische Quellcodegenerierung

Bei der automatischen Codegenerierung steht die Herstellung von Quellcode basierend auf einem Modell im Vordergrund. Betroffen sind von diesem Ansatz Teile der Systeme ASAP und APS. Da die automatische Codegenerierung, wie sich im Laufe der Arbeit herausstellte, mit erheblichem Aufwand verbunden ist, wurde auf ihre Anwendung auf die Systeme in ihrer Gesamtheit verzichtet. Der Kosten/Nutzen-Faktor liegt nur dann im akzeptablen Bereich, wenn Abschnitte des Quellcodes nach sich ähnelnden Mustern (z.B. Verarbeitung von Kommandos und Telemetrie) aufgebaut sind. ATUS hingegen wird, da die Testfunktionalitäten zu individuell sind, ausschließlich auf konventioneller Weise programmiert (d.h. ohne automatische Codegenerierung).

Insgesamt gibt es zwei verschiedene Programmiersprachen und Plattformen, für die Quellcode zu erstellen ist:

- C# auf dem .NET-Framework (Laufzeitumgebung auf einem Windows 8.1 Betriebssystem)
- C++ auf dem MicroBlaze mit RODOS (Betriebssystem auf einem Microprozessor)

Des Weiteren muss noch zwischen der Art von Software differenziert werden: Im Modell (siehe Abschnitt 7.5.1) wurden sowohl statische Strukturen als auch dynamisches Verhalten sowie die eigentlichen Daten beschrieben. Für jede dieser Arten gibt es verschiedene Vorgehensweisen, um aus dem zugrunde liegenden Modell Quellcode zu generieren:

- „Statische Strukturen“ können relativ einfach erstellt werden, indem die in der SysML definierten Blöcke (z.B. Activity Definition) aus dem Modell mit Hilfe einer Transformation (Skript) durch das Tool Enterprise Architekt in Klassendefinitionen für C# (als partielle Klasse) und C++ (als Klassen-Header) abgeleitet werden. Attribute werden entsprechend als Properties/Felder (C#) bzw. Member (C++) erstellt. Funktionen werden (vorerst) lediglich deklariert und durch dynamisches Verhalten zu einem späteren Zeitpunkt definiert. Die Transformation statischer Strukturen konnte ohne Probleme angewendet werden.
- Schwieriger hingegen erwies sich das „Dynamisches Verhalten“. Im Modell liegt dieses meist in Form von Aktivitäts- und Zustandsdiagrammen (SysML) vor. Zwar lassen sich aus diesen, ebenfalls relativ einfach durch Enterprise Architekt, standardisierte Implementierungen ableiten, jedoch sind meistens im Anschluss manuelle Anpassungen notwendig. Der Kosten/Nutzen-Faktor liegt daher nicht in einem akzeptablen Bereich. Daher wurde in der Regel auf eine automatische Codegenerierung des dynamischen Verhaltens verzichtet, und stattdessen eine konventionelle Implementierung „von Hand“ präferiert.
- Die eigentlichen Daten (z.B. Konfiguration von ASAP bzw. APS) sind auf Grund ihres Charakters nicht Gegenstand einer (direkten) automatischen Quellcodegenerierung. Sie werden jedoch im Modell (siehe Abschnitt 7.5.1) gepflegt und verwaltet. Dessen Infrastruktur (z.B. Datenbank und Datenmodell) hingegen wird automatisch erstellt. Daher kann von einer indirekten Quellcodegenerierung ausgegangen werden.

Die nachfolgende Tabelle 7-22 zeigt den Realisierungsgrad der automatischen Codegenerierung sowohl bzgl. der Systeme als auch der Arten der Codegenerierung. Die subjektiven Wertungen durch den Autor reichen von [+++] (sehr gut/vollständig), [++] (gut/überwiegend), [+] (befriedigend/teilweise), [0] (ausreichend/vereinzelt) bis [x] (nicht angewendet).

	Statische Strukturen	Dynamisches Verhalten	Daten
APS	+++	+	+++
ASAP	++	0	++
ATUS	x	x	+

Tabelle 7-22: Realisierungsgrad der automatischen Codegenerierung

7.5.3 Automatische Dokumentationsgenerierung

Eine nicht zu vernachlässigende Aufgabe im Rahmen der Systementwicklung ist die Dokumentation. Sie ist zum Gelingen einer Mission unbedingt erforderlich. Folgende Punkte zeichnen eine gute Dokumentation aus:

- eine gute Strukturierung der Inhalte
- die Abwesenheit von Fehlern
- Konsistenz der Daten (zwischen Implementierung und Dokumentation)
- die durchgehende Wahrung der Darstellungsform über das gesamte Dokument

Während der erste Punkt in den Verantwortungsbereich des Autors der Dokumentation fällt, kann bei den nachfolgenden Punkten die MBSE unterstützend mitwirken. Konkret wurde dazu die in Abbildung 7-100 dargestellte Vorgehensweise angewendet.

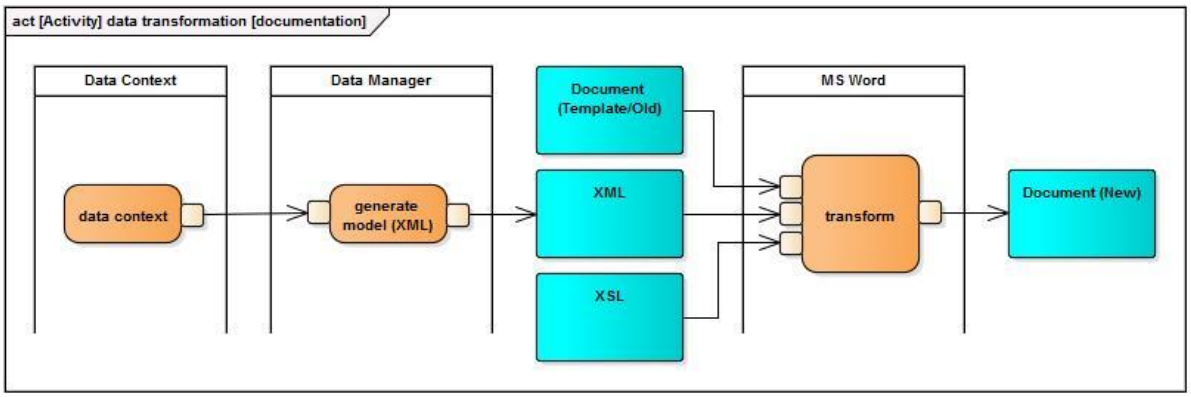


Abbildung 7-100: Datentransformation (Dokumentation)

Im Wesentlichen findet wieder eine Transformation von Daten statt. Die Roh-Daten werden zu diesem Zweck zuerst von einer Anwendung, dem Datenmanager, abgerufen. Dieser greift dazu auf den bereits eingeführten Daten-Kontext (siehe Abschnitt 7.5.1) zu. Die eigentliche Transformation bedient sich der Technologien XML (Extensible Markup Language) und XSL (Extensible Stylesheet Language). Beide arbeiten mit den Roh-Daten, unterscheiden sich aber in ihrem Verwendungszweck. Während erstere ausschließlich der Speicherung jener Roh-Daten dient, ist letztere für die eigentliche Transformation und der damit verbundenen Generierung des Output, in diesem Fall die Dokumentation als ein Microsoft Word Dokument, zuständig. Zum Beispiel wird in der XML-Datei ein Satz an Online-Telemetrie definiert. Dass diese Informationen in der Dokumentation mit einer bestimmten Formatierung in einer Tabelle darzustellen sind, kann der XSL-Datei entnommen werden. Diese Datei beschreibt, in welche Anweisungen die Roh-Daten umzuwandeln sind. Die Anweisungen müssen dementsprechend in der Zielsprache des Outputs (Dokument) geschrieben sein. Im vorliegenden Fall, also bei einem Microsoft Word Dokument, wäre das die WordProcessingML. Der Gestalter bestimmt im Dokument (MS Word) den Ort der Tabelle, fügt zu diesem Zweck ein Feld vom Typ INCLUDETEXT ein, und verlinkt diesen mit der XML- und der XSL-Datei. Bei der anschließenden Auswertung des Feldes wird der entsprechende Inhalt, im Beispiel die Tabelle, generiert. Bei späteren Änderungen der zugrunde liegenden Roh-Daten müssen die entsprechenden Felder des dann bereits existierenden Dokuments neu ausgewertet werden.

Der Vorteil dieser Vorgehensweise liegt darin, dass die Gestaltung des Dokuments zunächst wie gewohnt mit MS Word vorgenommen werden kann. Die Daten werden erst in einem weiteren Schritt per Transformation eingebettet. Damit ist eine klare Trennung zwischen Daten und Darstellung bzw. Präsentation gegeben.

Zu den zwei wichtigsten Dokumenten, die im Rahmen der vorliegenden Arbeiten auf diese Weise erstellt wurden, gehören:

- ICD-Dokument (Interface Control Document)
  Ein Datenblatt des zu beschreibenden Systems (z.B. ASAP) mit einem besonderen Schwerpunkt auf dessen Schnittstellen. Im Grunde genommen die Daten aus dem im Abbildung 7-99 dargestellten Datenmodell. Dazu gehören insbesondere Kommandierung und Telemetrie.
- Szenarien-Dokument
  Ein Dokument, das die im Abschnitt 7.1 genannten Referenzszenarien in konkreter Form beschreibt. „Konkret“ bedeutet in diesem Fall, dass für das APS eine entsprechende Konfiguration (z.B. Aktivitäten, Ressourcen) mit realen Werten vorliegt.

8 Ergebnisse

Die bisherigen Kapitel behandelten im Wesentlichen lediglich Entwurfsaspekte, die zwar insbesondere im Abschnitt 7 (Realisierung) detailliert auf einzelne Design-Aspekte des zu entwickelnden autonomen Planungssystems eingehen, jedoch ohne die tatsächliche Umsetzung bzw. dessen Stand und Fortschritt auf Systemebene näher zu erläutern. Dies soll nun im vorliegenden Kapitel erfolgen. Zu Beginn wird auf die verwendete Entwurfsmethode (siehe Kapitel 5 und Abschnitt 7.5) eingegangen. Anschließend wird das zu entwickelnde Planungssystem im Rahmen einer Evaluierung genauer auf seine Leistungsfähigkeit untersucht. Gefolgt von einer aktuellen Einschätzung bzgl. der Integration von ASAP in SONATE. Zum Abschluss des Kapitels werden die in den Abschnitten 7.2.3, 7.3.3, 7.4.3 gestellten Anforderungen auf ihre Erfüllung überprüft.

8.1 Entwurfsmethode

Die Erstellung eines Modells (siehe Abschnitt 7.5.1) gehört zu den Kernaufgaben der MBSE und ist zuerst mit einem Mehraufwand verbunden. Die statische Struktur, das dynamische Verhalten und die damit in Beziehung stehenden Daten konnten erfolgreich modelliert werden. Die Ergebnisse, überwiegend dargestellt in Form von SysML-Diagrammen wurden in der vorliegenden Arbeit durchgängig verwendet. Dieser Arbeitsschritt wird in anderen (konventionelleren) Entwicklungsmethoden vermieden. Dennoch hat sich der Autor bewusst für die MBSE entschieden. Die damit erstellten Entwürfe besitzen nämlich einen höheren Reifegrad, da die MBSE mächtige Beschreibungssprachen und Notationen verwendet. Hierdurch lassen sich kosten- und zeitintensive Entwurfsiterationen vermeiden.

Die automatische Codegenerierung (siehe Abschnitt 7.5.2) konnte nicht im gewünschten Umfang umgesetzt werden. Zwar ließen sich bei den statischen Strukturen akzeptable Ergebnisse erzielen, in Bezug auf das dynamische Systemverhalten sind die Ergebnisse jedoch nicht zufrieden stellend. Die automatische Quellcodegenerierung soll aber nicht generell als impraktikabel verworfen werden. Der Autor empfiehlt, im Rahmen künftiger Forschungsarbeiten diesen Aspekt weiter zu untersuchen und anzustreben, den Kosten/Nutzen-Faktor zu verbessern.

Die automatische Generierung von Dokumentationen (siehe Abschnitt 7.5.3) erwies sich, subjektiv durch den Autor bewertet, als sehr hilfreich. Besonders hervorzuheben ist der Effekt, dass auf Grund der Verwendung von Templates die Dokumentationen durchgehend einheitlich sind, und Daten desselben Typs immer in der gleichen Form dargestellt werden. Dadurch erhalten die Dokumente ein professionelleres Aussehen.

8.2 Autonome Planung

Im Rahmen der Evaluierung wurden die Funktionalität und die Leistungsfähigkeit des Autonomen Planungssystems (siehe Abschnitt 7.2) mit Hilfe der ATUS – Testumgebung erprobt und, wenn möglich, gemessen. Dieser Abschnitt fasst die Ergebnisse zusammen. Die detaillierten Messprotokolle sind dem Anhang beigefügt. Die Konfiguration des Autonomen Planungssystems orientiert sich an den im Abschnitt 7.1 beschriebenen Referenzszenarien und ist deshalb entsprechend den Angaben im Abschnitt 7.3.5.3.3 gesetzt.

Im Wesentlichen bestehen die Ergebnisse aus den durch das Planungssystem generieren Plänen. Sie werden exemplarisch anhand mehrerer Durchläufe in den folgenden Unterabschnitten diskutiert. Jeder Durchlauf hat zum Ziel eine Fähigkeit (Feature) des Planungssystems hervorzuheben. Um die Auswertung überhaupt zu ermöglichen, wurden in den Planungsalgorithmen Mechanismen zum

Protokollieren (sogenannte Logeinträge) eingebaut, die den Fortschritt und die Änderungen bei der laufenden Erstellung von Plänen festhalten.

Jeder Unterabschnitt beginnt mit einer Darstellung der Ergebnisse in tabellarischer Form. Zu den Angaben gehören:

- Index: Jedes Inkrement des Indexes entspricht einer Planänderung bzw. einem Logeintrag
- Nummer des Plans: Dient als Referenzschlüssel zur Unterscheidung der jeweiligen Pläne
- Punkte des Plans: Eine Maßzahl, die die Qualität des Plans ausdrückt.
- Änderungen am Plan: Beschreibt, welche Änderungen (z.B. Einfügen neuer Aktivitäten) am Plan vorgenommen wurden (siehe Abschnitt 7.2.5.5).
- Prozessorzeit: Gibt an, wieviel Prozessorzeit benötigt wurde, um die Planänderungen (nur Bindungen und Erwägungen) zu berechnen. Die Messungen wurden auf einem PC mit einem 4-Kernprozessor mit jeweils 4.0 GHz durchgeführt. Effektiv verwendet der Ablaufplaner jedoch nur einen einzigen Kern.

An dieser Stelle soll vermerkt werden, dass aus Gründen der Übersicht nicht jeder einzelne Schritt der Planung genau betrachtet wird, sondern lediglich eine Auswahl (siehe Index).

Im Anschluss an die Tabelle werden die einzelnen Zwischenergebnisse der Planung, d.h. die Pläne, in graphischer Form dargestellt. Dabei wird der zeitliche Verlauf der einzelnen Aktivitäten und Ressourcen sichtbar.

Die Ergebnisse zeigen, dass die Ablaufplanung in der Lage ist, die Pläne fortlaufend zu verbessern (siehe Punkte in der Tabelle), und dabei die Nebenbedingungen, gegeben durch zeitliche Beschränkungen der Aktivitäten und durch Wertebeschränkungen (Min, Max) der Ressourcen, einhält.

Die zum Plan zugehörigen Kommandos werden, vereinfacht betrachtet, durch direkte Abbildung (siehe Abschnitt 7.2.5.6) der Aktionen (von Aktivitäten) abgeleitet. Sie sind ebenfalls in den Messergebnissen im Anhang aufgelistet.

8.2.1 Deterministische Entscheidungsumsetzung ohne Optimierung

Diese Messreihe zeigt den typischen Ablauf bei der Inbetriebnahme des Autonomen Planungssystems. Es wird davon ausgegangen, dass eine gewisse Anzahl an deterministischen Aktivitäten, z.B. die Aufnahme von Bildern oder Videos mit Hilfe der in ASAP verbauten Kamera, in Auftrag gegeben wurde. In diesem Fall sind das die Aktivitäten „Search“ (1x), „Image“ (3x) und „Video“ (2x). Aufgrund der Beschränkung der Ressourcen und der deaktivierten Optimierung war das Planungssystem lediglich in der Lage, die ersten vier Aktivitäten erfolgreich zu planen.

Zeit	**Index (Logeintrag)**	**Nummer (Plan)**	**Punkte (Plan)**	**Änderungen (Plan)**	**Prozessorzeit [hh:mm:ss ms]**
00:00:04	1	77	4.600	Place 1	00:00:00 045
00:00:05	2	79	6.900	Place 2	00:00:00 056
00:00:06	3	81	9.200	Place 3	00:00:00 081
00:00:06	4	83	11.500	Place 4	00:00:00 145
	>= 5		11.500	-	

Tabelle 8-1: Deterministische Entscheidungsumsetzung ohne Optimierung (Messreihe A)

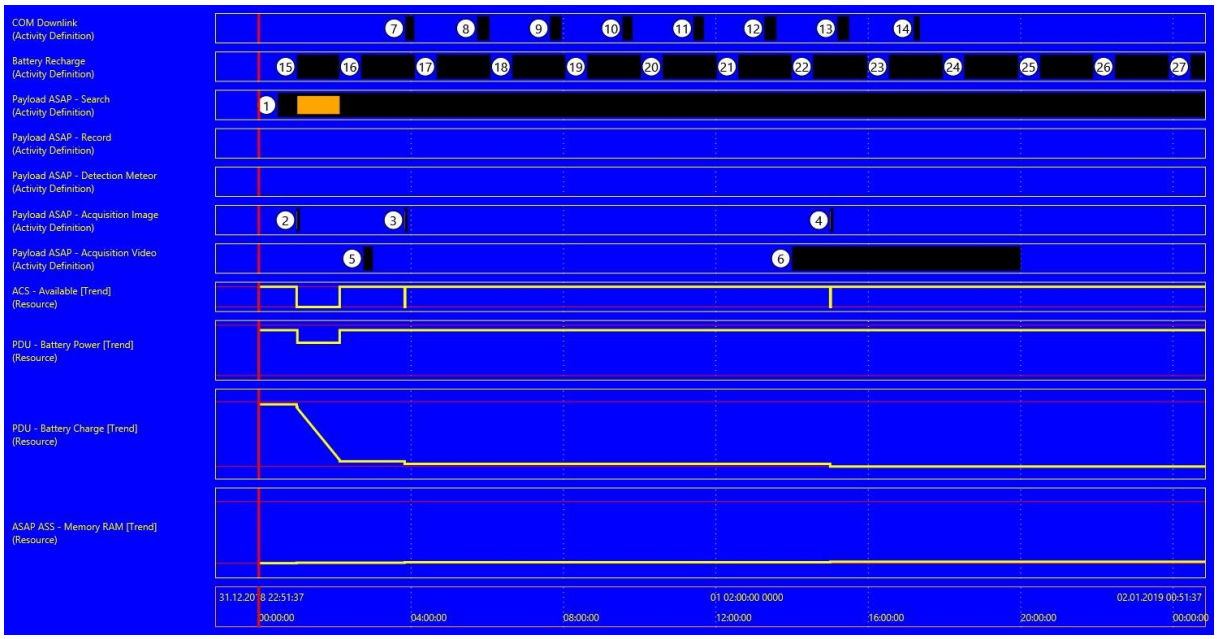


Abbildung 8-1: Deterministische Entscheidungsumsetzung ohne Optimierung (Messreihe A Index 4)

8.2.2 Anschließende Optimierung (nach Entscheidungsumsetzung)

Im Anschluss an die Messreihe A (Abschnitt 8.2.2) wurde die Optimierung aktiviert. Der bis dahin bestehende Plan wurde hierbei verbessert, indem Aktivitäten zum Auffüllen der Batterien (Aktivitäten 15 und 16) aufgenommen wurden und die Suche (Aktivität 1) unterbrochen wurde. Mit den dadurch freiwerdenden Ressourcen konnten weitere Videosequenzen (Aktivitäten 5 und 6) geplant werden.

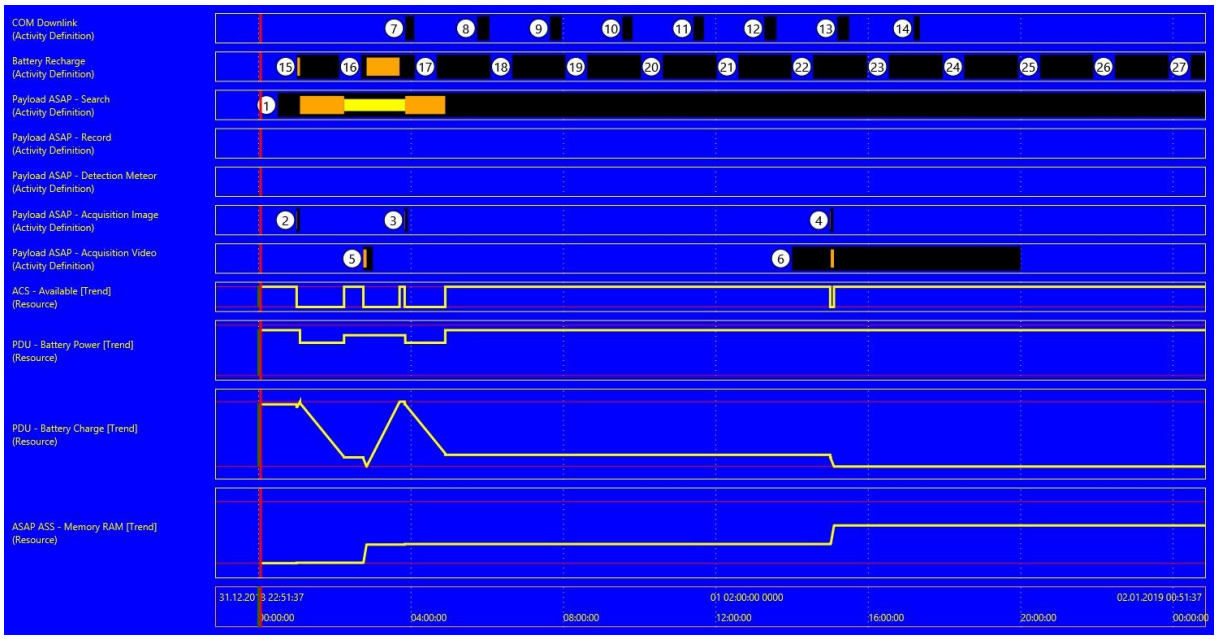


Abbildung 8-2: An vorherige Entscheidungsumsetzung anschließende Optimierung (Messreihe B Index 13)

Zeit	Index (Logeintrag)	Nummer (Plan)	Punkte (Plan)	Änderungen (Plan)	Prozessorzeit [hh:mm:ss ms]
00:00:19	5	120	11.900	Place 15	00:00:02 218
00:00:21	7	122	12.300	Place 16	00:00:01 984
00:01:25	10	125	16.840	Passive 1	00:00:54 266
00:01:48	13	128	26.040	Place 5 Place 6	00:00:55 154

Tabelle 8-2: An vorherige Entscheidungsumsetzung anschließende Optimierung (Messreihe B)

8.2.3 Gleichzeitige Entscheidungsumsetzung und Optimierung

Im Gegensatz zu den Messreihen A und B, bei denen die Vorgänge „Umsetzung“ und „Optimierung“ separat und sequentiell abliefen, wurde in Messreihe C eine gleichzeitige Ausführung vorgenommen. Auffällig hierbei ist, dass, trotz gleicher Eingangsdaten, die Qualität (d.h. der Punktestand) der Messreihe C schlechter ist als bei A und B. Hieraus folgt, dass entweder das Autonome Planungssystem eine zusätzliche Funktionalität benötigt, die „intelligent“ zwischen „Umsetzung“ und „Optimierung“ schaltet, oder, dass der Scheduling-Algorithmus im speziellen bezgl. dieses Aspektes verbessert werden muss.

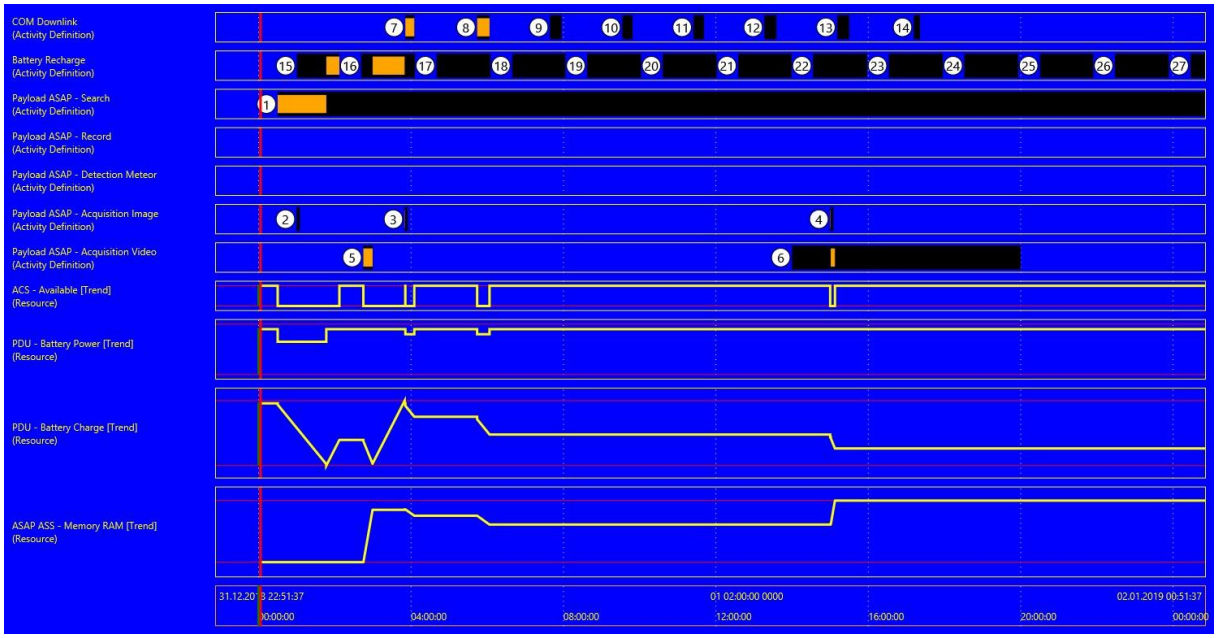


Abbildung 8-3: Gleichzeitige Entscheidungsumsetzung und Optimierung (Messreihe C Index 15)

Zeit	Index (Logeintrag)	Nummer (Plan)	Punkte (Plan)	Änderungen (Plan)	Prozessorzeit [hh:mm:ss ms]
	…	… (139)	11.500	Place 1, 2, 3, 4	…
00:00:08	5	145	13.400	Place 5 Unplace 1	00:00:01 109
	…	… 186	18.747	Place 6 Unplace 2 Place 2,7,8, 15, 16	…
00:02:35	15	189	20.854	Place 1 Unplace 2	00:00:04 972

Tabelle 8-3: Gleichzeitige Entscheidungsumsetzung und Optimierung (Messreihe C)

8.2.4 Reparatur ungültiger Pläne

Ausgangssituation ist die Messreihe C aus dem Abschnitt 8.2.3. Die vorliegende Messreihe soll zeigen, dass das Autonome Planungssystem in der Lage ist, ungültig gewordene Pläne zu reparieren. In diesem Fall wurde in der dreizehnten Minute ein Einbruch des verfügbaren RAM Speichers emuliert. Die Folge hieraus war, dass der ursprünglich berechnete Plan aufgrund einer Verletzung der Ressourcen-Beschränkung ungültig wurde. Die Reparatur-Funktionalität hat in diesem Fall die Aktivität 5 aus dem Plan entfernt, um die Gültigkeit wiederherzustellen. Dieser Vorgang hatte eine Verringerung des Punktestands zur Folge.

Zeit	Index (Logeintrag)	Nummer (Plan)	Punkte (Plan)	Änderungen (Plan)	Prozessorzeit [hh:mm:ss ms]
00:02:35	15	189	20.854	Place 1 Unplace 2	00:00:04 972
00:13:08	16	191	14.955	Unplace 5	00:00:11 728

Tabelle 8-4: Reparatur ungültiger Pläne (Messreihe D16)

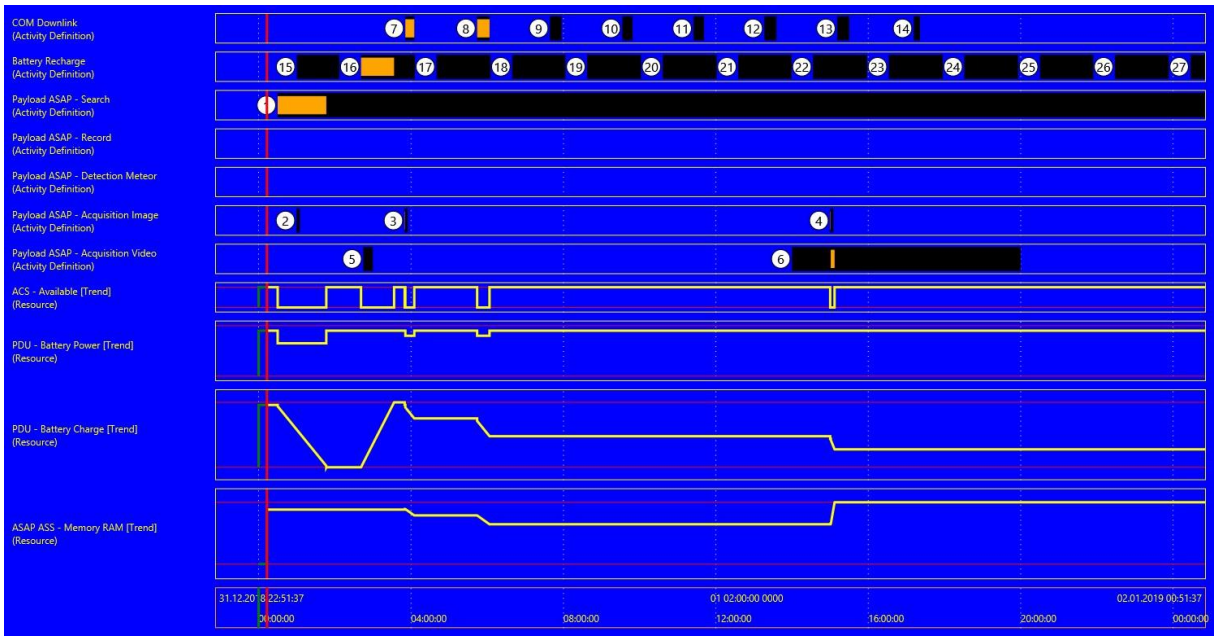


Abbildung 8-4: Reparatur ungültiger Pläne (Messreihe D16)

8.2.5 Gleichzeitige Planung und Ausführung von Plänen (Fixierung)

Die in den bisherigen Abschnitten 8.2.1 bis 8.2.4 gezeigten Funktionalitäten (Umsetzung, Optimierung und Reparatur) waren stets auf die Zukunft gerichtet, da keine Aktivität zum Zeitpunkt der Planung ausgeführt wurde. In diesem Unterabschnitt soll daher das Verhalten des Autonomen Planungssystems bei laufenden Änderungen untersucht werden. Die hierfür erstellte Messreihe E baut auf den Ergebnissen der Messreihe A auf.

Nach der initialen Berechnung (Messreihe A) wurde der Plan bis zum Zeitpunkt 01:25:17 mit ausgeschalteter Optimierung ausgeführt. Zu diesem Zeitpunkt befand sich Aktivität 1 bereits in der Ausführung, d.h. es wurden Kommandos generiert, mit dem Board-Computer des Satelliten synchronisiert und dort ausgeführt. Anschließend wurde die Optimierung aktiviert. Die

Messergebnisse zeigen, dass der Ablaufplaner den fixen Startzeitpunkt der Aktivität 1 respektierte und nur die Aktionen optimierte, die ab dem Zeitpunkt der Änderung in der Zukunft lagen.

Zeit	Index (Logeintrag)	Nummer (Plan)	Punkte (Plan)	Änderungen (Plan)	Prozessorzeit [hh:mm:ss ms]
00:00:06	4	83	11.500	Place 4	00:00:00 145
	…	…	…	Place 16	
01:25:28	8	123	11.955	Passive 1	00:00:09 857
01:25:28	9	124	13.957	Adjust	00:00:00 664
01:25:31	10	125	19.474	Place 5 Place 6	00:00:02 796

Tabelle 8-5: Gleichzeitige Planung und Ausführung von Plänen (Messreihe E10)

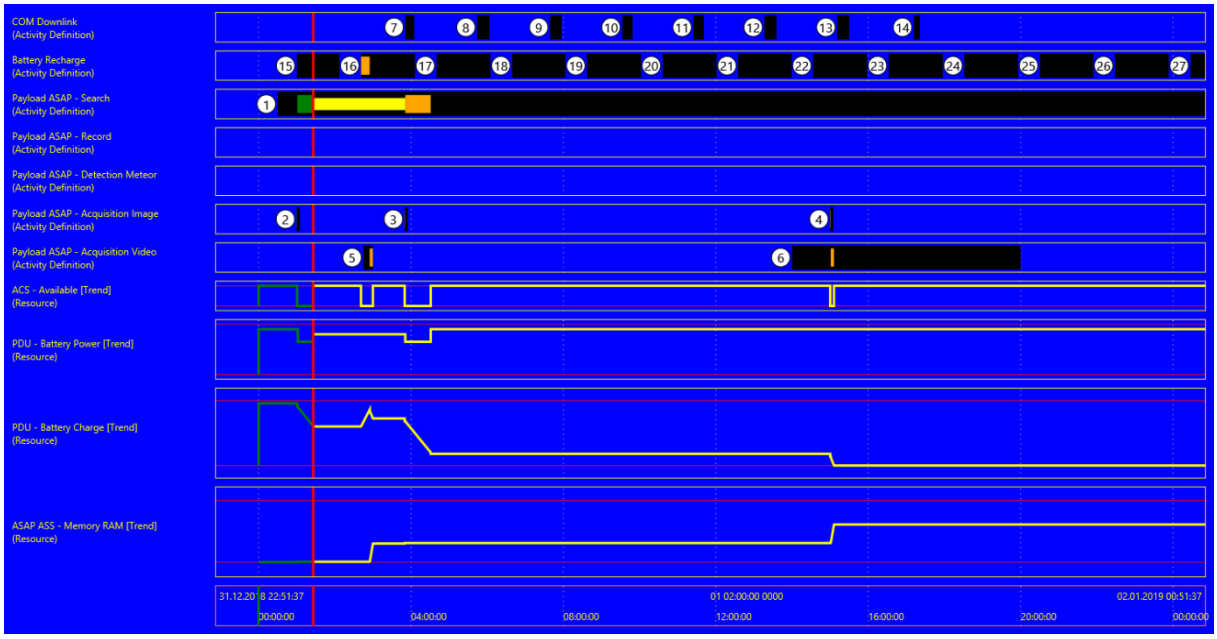


Abbildung 8-5: Gleichzeitige Planung und Ausführung von Plänen (Messreihe E10)

8.2.6 Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität

Die letzte Messreihe zeigt die Fähigkeit des Autonomen Planungssystems, auf nicht vorhersehbare kurzfristige Ereignisse schnell reagieren zu können. Ausgangssituation war die Messreihe B aus dem Abschnitt 8.2.2. Der Plan wurde bis zum Zeitpunkt 00:20:56 wie ursprünglich vorgesehen abgearbeitet. In diesem Moment jedoch trat ein entsprechendes Ereignis (z.B. Sichtung eines Meteors) ein. Das Planungssystem erzeugte auf diesen Stimulus hin innerhalb kürzester Zeit (in diesem Fall einige Millisekunden) einen neuen Plan, der die Beobachtung des Ereignisses (Aktivität 39) vorsah. Dadurch wurde eine sofortige Synchronisierung der Kommandolisten ausgelöst. Im weiteren Verlauf wurde die Aktivität 39 immer wieder verlängert (nur solange das Ereignis noch aktiv blieb), um den Planungsoutput zu maximieren.

Zeit	Index (Logeintrag)	Nummer (Plan)	Punkte (Plan)	Änderungen (Plan)	Prozessorzeit [ms]
00:01:48	13	128	26.040	Place 5, 6	00:00:55 154

00:20:32	14	131	10.600	Place 39 Unplace 1, 2, 3, 4 Unplace 5, 6, 7 Unplace 15, 16	00:00:00 170
	…			Adjust	
00:20:56	19	141	10.840	Adjust	00:00:00 069

Tabelle 8-6: Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität (Messreihe F)

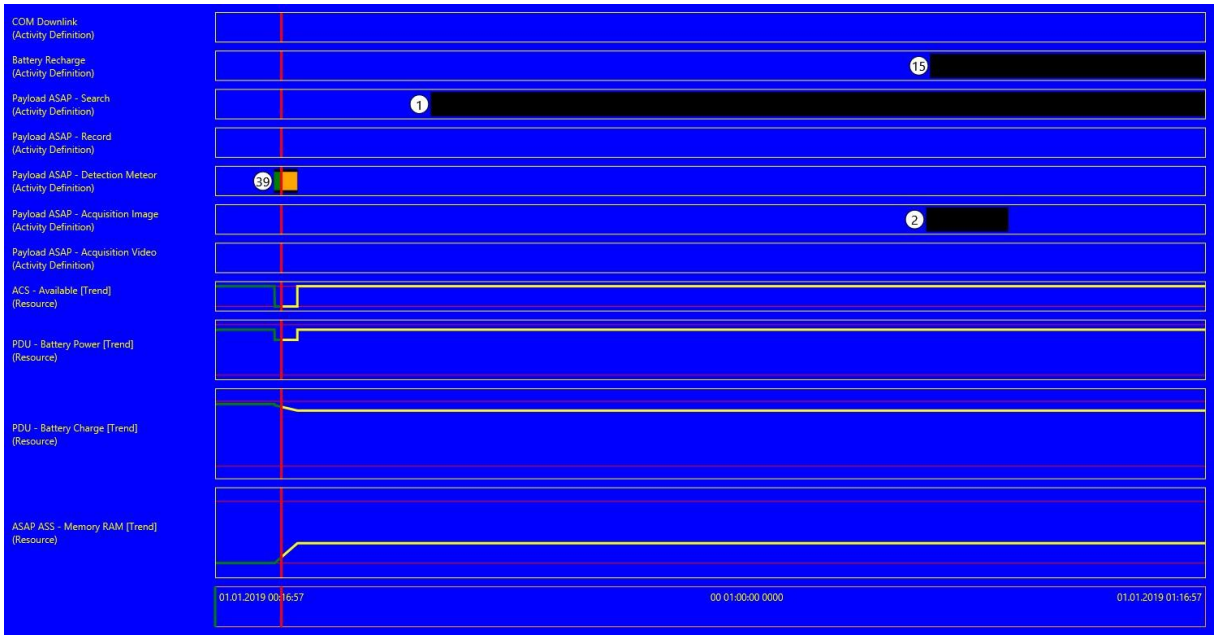


Abbildung 8-6: Nichtdeterministische Entscheidungsumsetzung mit hoher Priorität (Messreihe F19)

8.3 Geplanter Betrieb von ASAP in SONATE

Zum Zeitpunkt der Fertigstellung der vorliegenden Dissertation befand sich das Vorhaben SONATE in der Design-Phase (Phase C). Eine technisch reduzierte Variante von ASAP soll als experimentelle Nutzlast in diesem Satelliten eingesetzt und erprobt werden. Das bisherige Konzept (Stand 4.Quartal 2016) bzgl. der Einbindung von ASAP in die Missionsplanung von SONATE soll, da es ebenfalls einen Realisierungsstand von ASAP darstellt, in seinen Grundzügen in diesem Abschnitt vorgestellt werden. Die Erläuterungen werden sich ausschließlich auf den geplanten Missionsbetrieb, die Kommandierung und die Telemetrierung beschränken.

ASAP ist in SONATE eine experimentelle Nutzlast und dient dem Zweck, die autonomen Funktionalitäten zu erproben. Dementsprechend sind die einzelnen Phasen des Missionsbetriebes im Experiment- und Betriebsszenario angepasst:

- Commissioning-Phase
  Die Inbetriebnahme dient dazu, sämtliche Hardware- und Softwarefunktionen von ASAP zu testen und in Einsatzbereitschaft für die anderen Betriebsphasen zu versetzen. Dafür sind folgende Handlungen vorgesehen:
  - Test der allgemeinen Kommandierung und Telemetrierung von ASAP
  - Test der Sensoren
  - Test der verschiedenen Speicher (SRAM, DRAM, Flash)
  - Test der Bildsensoren
  - Test der Datenübertragung zum Board- und zum Nutzlastcomputer

  - Test spezieller Hardware (z.B. Watchdog)
  - Test der autonomen Sensorplattform (ASAP-ASS), nur Grundfunktionalität
  - Test des autonomen Planungssystems (ASAP-APS), nur Grundfunktionalität
- Operation-Phase
  In der Operation-Phase werden die einzelnen Teilnehmer der autonomen Planung, also der Board-Computer (OBDH), das autonome Planungssystem (ASAP APS) und die autonome Sensorplattform (ASAP ASS), individuell je nach Betriebsmodus ab- bzw. zugeschaltet und entsprechend der Experiment- und Betriebsszenarien betrieben. Die Szenarien sind den in Abschnitt 7.1 beschriebenen Referenzszenarien angelehnt, allerdings mussten Anpassungen vorgenommen werden, da SONATE aufgrund seiner eingeschränkten Lageregelung zu den ursprünglichen Manövern nicht fähig wäre.

An Bord von SONATE sind neben den Sub-Systemen des Bussystems auch mehrere Nutzlasten integriert. Zu diesen gehört auch ASAP. Daher existieren viele Kommunikationsteilnehmer. Dementsprechend muss der Fluss an Telemetrie und Telekommando geregelt werden. Dazu wird zwischen vier Typen von Telemetrie unterschieden. Die Einteilung wurde vom Herrn Oleskii Balagurin (Projektleiter von SONATE) wie folgt vorgegeben:

- Online-Telemetrie (im Abschnitt 2.2.4.3 als Real-Time-Telemetrie bezeichnet)
  Ein Satz der wichtigsten Telemetrie-Werte (Housekeeping) aller Sub-Systeme/Nutzlasten des Satelliten. Die Online-Telemetrie wird nur während eines Kontaktes mit der Bodenstation vom Satelliten erfasst und gesendet. Die Aktualisierungsrate ist relativ hoch (< 5 Sekunden).
- Offline-Telemetrie (im Abschnitt 2.2.4.3 als History-Telemetrie bezeichnet)
  Derselbe Satz wie Online-Telemetrie, jedoch mit dem Unterschied, dass die Erfassung auch außerhalb der Kontaktzeiten mit der Bodenstation stattfindet. Die Aktualisierungsrate ist daher im Vergleich zur Online-Telemetrie geringer (ca. 60 Sekunden).
- Extended-Telemetrie
  Für jedes Sub-System und jede Nutzlast existieren eigene Sätze Telemetrie (nur Housekeeping), die bei Bedarf vom Operator angefordert werden können.
- Nutzlast-Telemetrie
  Reine Nutzlastdaten der einzelnen Nutzlasten im Satelliten.

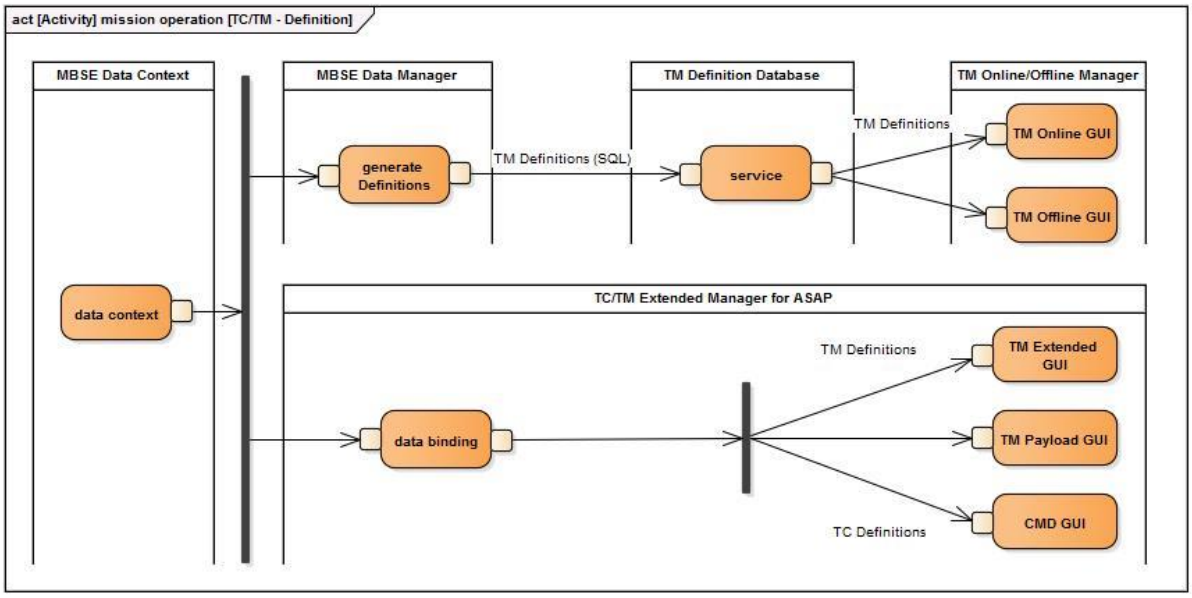


Abbildung 8-7: Missionsbetrieb von SONATE - Definition von Telekommandos und Telemetrie

Zur Handhabung der Kommandos und der Telemetrie müssen die involvierten Daten sowohl definiert als auch verarbeitet werden. Die Realisierung dieser beiden Aufgaben in SONATE, bezogen auf ASAP, wird im Folgenden, beginnend mit den Definitionen (siehe Abbildung 8-7), näher erläutert.

Für die Handhabung der Online/Offline-Telemetrie wurde unter der Leitung von Herrn Gerhard Fellinger ein spezielles Softwarepaket, bestehend aus einer graphischen Oberfläche und einer Datenbank, entwickelt. Die GUI, bzw. die darunterliegende Logik, greift auf die in der Datenbank abgelegten Definitionen zurück, um den eingehenden Datenstrom (ein reiner Bytestrom) an Telemetrie zu interpretieren und in spezifischer Form graphisch darzustellen. Im Grunde beschreiben die Definitionen Zuordnungen, bei denen Datensignale aus dem Datenstrom extrahiert, mit Hilfe einer Übersetzungsfunktion interpretiert, und die Ergebnisse auf eine bestimmte Position der GUI gesetzt werden. Für die Anbindung von ASAP an das bestehende Online-Telemetriesystem entschied sich der Autor, auf die bereits bestehende Infrastruktur der MBSE (siehe Abschnitt 7.5.1) zurückzugreifen. Das Datenmodell wurde dementsprechend mit den Konzepten der Telemetrie und der Kommandos erweitert. Hierdurch ergeben sich für ASAP die bereits bekannten Vorteile wie Datenkonsistenz und automatische Generierung von Quellcode und Dokumentation. Da im Falle von ASAP die Daten des Modells, und damit auch die Definitionen der Online/Offline-Telemetrie, auf einem anderen Datenbank-Server gehostet werden, müssen mit Hilfe einer Anwendung die beiden Datenbanken synchronisiert werden. Die Handhabung der Extended-Telemetrie von ASAP fiel ausschließlich in den Verantwortungsbereich des Autors. Dementsprechend konnte die Infrastruktur der MBSE direkt genutzt werden, indem die graphischen Benutzeroberflächen mittels Datenbindung auf den Datenkontext zugreifen.

Mit Hilfe der Definitionen kann nun die zweite Aufgabe – die eigentliche Handhabung von Telemetrie und Telekommandos – bewältigt werden. Der entsprechende Ablauf ist in der nachfolgenden Abbildung 8-8 dargestellt.

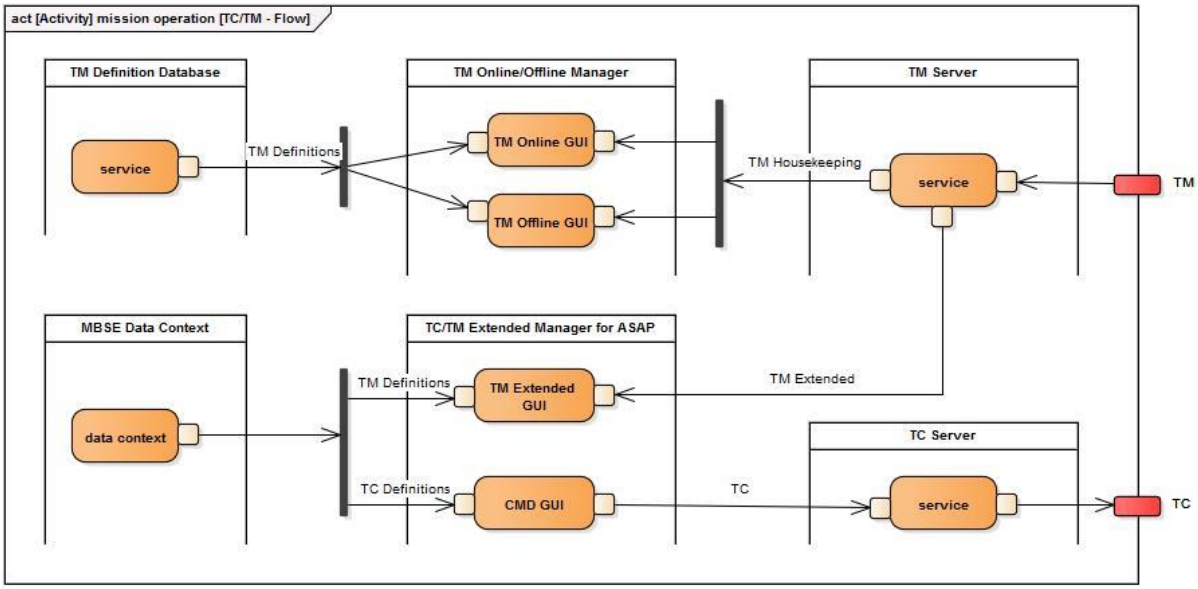


Abbildung 8-8: Missionsbetrieb von SONATE – Datenfluss von Telemetrie und Telekommando

Eingangsseitig nimmt ein TM-Server sämtliche Telemetrie entgegen. Die einzelnen Pakete dieses Datenstroms werden zuerst nach ihrem Typ (z.B. Online-Telemetrie oder Extended-Telemetrie von ASAP) getrennt. Es ist vorgesehen, diese Trennung konform der CCSDS-Spezifikation mit Hilfe der virtuellen Kanäle vorzunehmen. Online/Offline – Telemetrie wird dann zur bereits oben erwähnten GUI-Anwendung weitergereicht und mit Hilfe der Definitionen interpretiert und dargestellt. So ähnlich wird auch bei der Extended-Telemetrie vorgegangen. Der Unterschied besteht darin, dass hier das Ziel der Datenpakete dedizierte Anwendungen für das jeweilige Sub-System bzw. Nutzlast sind. Die

speziell für ASAP vorgesehene Anwendung (wurde bereits oben erwähnt) nimmt ebenfalls die von ASAP gesendete Extended- und Nutzlast-Telemetrie und interpretiert diese mit Hilfe der Definitionen.

8.4 Validierung der Anforderungen

Bei der Entwicklung der im Kapitel 7 (Realisierung) beschriebenen Systeme – APS, ASAP und ATUS – wurde die im Kapitel 5 erläuterte Entwurfsmethode angewendet. Zu Beginn wurden daher für jedes System Anforderungen (siehe Abschnitte 7.2.3, 7.3.3, 7.4.3) festgelegt, die im Rahmen der Evaluierung auf ihre Erfüllung überprüft wurden. Dieser Abschnitt zeigt die diesbezüglichen Ergebnisse. Hierbei werden nur Anforderungen aufgelistet, die nicht vollkommen erfüllt wurden. Dazu werden jeweils eine Begründung sowie der weitere Umgang mit der Abweichung genannt. Im Umkehrschluss sind in diesem Abschnitt nicht aufgeführte Anforderungen als vollkommen erfüllt zu betrachten.

APS:

Nummer	Name	Begründung	Vorgehensweise
2.3.2.1	TC Format	Eine genaue Spezifikation der Anforderungen (z.B. Protokoll) lag zum Zeitpunkt der Fertigstellung der vorliegenden Arbeit nicht vor.	Das APS verwendet eigene Protokolle und Formate. Sobald eine passende Spezifikation (gegeben durch SONATE) vorliegt, werden nachträglich Änderungen an den betroffenen Protokollen und Formaten vorgenommen.
2.3.2.2	TM Format		
2.5.1	Protokoll Kommandoverwaltung		
2.6.1	Protokoll Zustandserfassungs-verwaltung		

Tabelle 8-7: Validierung der Anforderungen an das APS

ASAP:

Eine genaue Verifikation der Anforderungen von ASAP war zum Zeitpunkt der Fertigstellung der vorliegenden Arbeit nicht möglich, da die Umsetzungsarbeiten an dem System noch nicht abgeschlossen waren. Es ist daher nicht auszuschließen, dass in Zukunft weitere Anforderungen die Verifikation nicht bestehen. Die nachfolgende Tabelle listet daher nur jene Anforderungen auf, die bis zum jetzigen Zeitpunkt Diskrepanzen aufweisen.

Nummer	Name	Begründung	Vorgehensweise
1.6.1	Software Update	Wegen der Größe des zu aktualisierenden Speicherabbildes (4 MB) und der für die Kommunikation (UHF/VHF) zur Verfügung stehenden Infrastruktur ist die benötigte Dauer für einen Aktualisierungsvorgang inakzeptabel.	Es ist zu untersuchen, ob sich das Problem mittels Datenkompression und dem Hochladen einzelner Datenfragmente lösen lässt.
4.4.2.2	Elektrische Leistung 10W	ASAP wird im nominellen Betrieb auf jeden Fall mehr als 10 W an elektrischer Leistung benötigen.	Die Anforderung wurde vom Lehrstuhl als Orientierungsgröße vorgegeben, und kann daher angepasst (z.B. 15 W) werden. Diese Maßnahme ist mit Bedacht zu wählen, da sie enorme Auswirkungen auf das Power Budget eines Satelliten hat.

Tabelle 8-8: Validierung der Anforderungen an ASAP

ATUS:

Nummer	Name	Begründung	Vorgehensweise
2.1.3.4	Kommunikations-anwendung Bodensegment	Eine genaue Spezifikation der Anforderungen (z.B. Protokoll) lag zum Zeitpunkt der Fertigstellung der vorliegenden Arbeit nicht vor.	Der Aspekt der Kommunikation wurde in der Simulation nicht berücksichtigt. Der Datenaustausch (Kommandos und Telemetrie) erfolgt direkt zwischen den Anwendungen des Bodensegments (Operator) und des Raumsegments (z.B. APS).
2.1.3.5	Kommunikations-anwendung Raumsegment		
2.1.4.5	HS-Bus		
2.1.3.8	Energieversorgungs-anwendung	Zeitlicher Mangel, niedrige Priorität bei der Umsetzung	Wird noch umgesetzt
2.1.4.3	I2C-Bus		

Tabelle 8-9: Validierung der Anforderungen an ATUS

9 Zusammenfassung

Satelliten in die Lage zu versetzen, autonome Entscheidungen zu treffen, ist eine technisch sehr anspruchsvolle Aufgabe. Diese bildet die Problemstellung der vorliegenden Arbeit. Falls sie gelöst wird, kann der Gewinn einer Satellitenmission, sei es wissenschaftlicher oder kommerzieller Art, gesteigert werden, indem Ressourcen besser genutzt und Abhängigkeiten in der Kommunikation zwischen Satellit und Bodenstation aufgehoben werden. Die vorliegende Arbeit leistet einen Beitrag zum aktuellen Stand der Forschung/Technik, indem sie eine Möglichkeit aufzeigt, die Problemstellung zu bewältigen. Im Mittelpunkt steht dabei die Entwicklung eines autonomen Planungssystems. Dieses kann mit der Onboard-Datenverwaltung des Satelliten, insbesondere der Kommandoverwaltung, interagieren und somit autonom Einfluss auf die Steuerung des Satelliten nehmen, ohne dass ein Eingreifen des Operators notwendig wäre.

Als Entwurfsmethode kam nach Möglichkeit die modellbasierte Systementwicklung (MBSE) zum Einsatz. Der Ansatz, ein Modell in den Mittelpunkt des Entwurfs zu stellen und daraus sämtliche Dokumente, wie zum Beispiel Dokumentation oder Quellcode, abzuleiten, scheint zwar sehr verlockend, da sich auf diese Weise eine Konsistenz sicherstellen lässt, stößt aber in seiner Umsetzung auf Hindernisse. Folglich wurden Teile der zu erstellenden Systeme auf klassischer Weise entwickelt.

Das autonome Planungssystem entstand im Rahmen von Vorhaben/Projekten. Dadurch bedingt steht die Arbeit in einem Kontext, nämlich der Detektion kurzfristiger Ereignisse und Phänomene (z.B. Blitze) in der Erdatmosphäre. Damit verbunden ist ein Satz von Anforderungen, die das zu entwickelnde autonome Planungssystem zu erfüllen hat. Darüber hinaus wurde es so konzipiert, dass es auch in anderen Satellitenmission eingesetzt werden könnte. In dieser Anpassbarkeit mittels Konfiguration sticht das autonome Planungssystem hervor und liefert somit einen wesentlichen Beitrag zum Stand der Technik und der Forschung.

Um seinen Anforderungen gerecht zu werden, muss das autonome Planungssystem mehrere Aufgaben erfüllen:

- Entscheidungsfindung
- Ablaufplanung
- Planausführung und Synchronisation mit der Kommandoschnittstelle des Satelliten
- Daten- und Ereignisaufbereitung

Die dabei entstehenden Pläne, bestehend aus Aktivitäten, Aktionen, Ressourcen (Verlauf) und Kommandos, gilt es, unter Einhaltung zeitlicher und ressourcenbeschränkender Nebenbedingungen zu optimieren.

Das autonome Planungssystem ist eine reine Softwarekomponente, die in ein umspannendes System einzubetten ist. Dieses muss zum einen die zur Ausführung benötigten Hardware-Ressourcen (z.B. Prozessor und Speicher) zur Verfügung stellen, zum anderen die erforderlichen Schnittstellen anbieten.

Im Falle der vorliegenden Arbeit übernimmt das System ASAP die oben beschriebene Aufgabe. Hierbei handelt es sich um ein optisches Kamerasystem mit autonomen Fähigkeiten. Es ist mit einer Sensorplattform ausgestattet, mit der die Umgebung beobachtet werden kann. Ihre leistungsstarke Hardware und die darin implementierten bildverarbeitenden Algorithmen können selbst Ereignisse und Phänomene mit einer Lebensdauer von einigen Millisekunden (z.B. Blitze in der Erdatmosphäre) selbständig detektieren und klassifizieren. Aus diesen Rohdaten werden aussagekräftige Meta-Daten generiert, die als Input für das nachgeschaltete autonome Planungssystem dienen. Dieses hat die Aufgabe, entsprechend seiner Konfiguration, die sich an vorgegebenen Missionsszenarien orientiert, Pläne (insbesondere die Kommandos) zu generieren und mit der Bordsteuerung des Satelliten zu synchronisieren.

Der zuletzt genannte Schritt, also die Interaktion zwischen dem autonomen Planungssystem (bzw. ASAP) und der Bordsteuerung des Satelliten, musste simuliert werden, da zum Zeitpunkt der Entwicklung des autonomen Planungssystems und ASAP kein Satellit bzw. keine mit der autonomen Planung involvierten Komponenten des Satellitenbusses, zur Verfügung standen. Zu diesem Zweck wurde ein Testumgebungssystem (ATUS) entwickelt, das die notwendigen Schnittstellen bereit stellt und das Verhalten des Satelliten entsprechend simuliert.

10 Ausblick

Die vorliegende Arbeit ist eine Etappe auf dem langen Weg hin zur effizienten und sicheren autonomen Planung. Im Rahmen der involvierten Vorhaben wurden mit den Systemen APS und ASAP grundlegende Bausteine geschaffen, mit denen sich autonome Handlungen planen lassen. Jedoch hatten beide Systeme bisher noch nicht die Gelegenheit, sich im Rahmen einer realen Satellitenmission in der Praxis zu bewähren. Die nächste Möglichkeit bietet sich mit dem Vorhaben SONATE. Eine hardwaretechnisch reduzierte Variante von ASAP, die dennoch unter anderem mit dem APS ausgestattet ist, ist für diesen Satelliten vorgesehen. Hierzu laufen derzeit Anpassungsarbeiten, um die Integration der autonomen Systeme in den Satelliten zu ermöglichen.

Ferner ist angedacht, den Funktionsumfang vom APS zu erweitern. Da diese Absicht zu Beginn des Entwurfsprozesses berücksichtigt wurde, verfügt das System über einen modularen Aufbau und über klar definierte Schnittstellen zwischen den einzelnen Teilkomponenten. So könnten neue Arten von Entscheidungskomponenten, die zum Beispiel zielbasiert oder auf Basis von neuronalen Netzwerken arbeiten, entwickelt und in den Planer integriert werden. Eine weitere vielversprechende Möglichkeit bietet das autonome Lernen, indem das durch die Konfiguration vorgegebene Verhaltensmodell (z.B. die Auswirkungen von Aktivitäten auf Ressourcen) des APS mit dem tatsächlichen Verhalten des Satelliten zur Laufzeit verglichen und angepasst wird. Es fände sozusagen eine Selbstkorrektur statt. Dadurch ließen sich die zur Verfügung stehenden Ressourcen besser nutzen. Einen weiteren Ansatz bietet die Ablaufplanung. Der verwendete Scheduling-Algorithmus hat eine eher intuitive, an die menschliche Denkweise orientierte Zielfindung (Abwägen, Schätzen, Binden). Eine Untersuchung der Algorithmen aus dem Bereich der (nicht-linearen) Optimierung könnte vielleicht einen besseren Ansatz für die Ablaufplanung aufzeigen. Globale Optima (d.h. die Berechnung des besten Plans) ließen sich dadurch leichter finden.

Ein weiterer Aspekt betrifft die Sicherheit des Satelliten. Im Abschnitt 7.2.6.2 wurde bereits auf mögliche Gefahrenquellen sowie auf Maßnahmen zur Förderung der Sicherheit eingegangen. Da sich die autonome Planung generell in der Forschung befindet, ist das bisherige Sicherheitskonzept (SafeMode) zum Zweck der Technologieerprobung vertretbar. Ein Einsatz der autonomen Planungssysteme in größeren und/oder kommerziellen Satellitenmissionen ist nach dem bisherigen Stand der Technik nach Meinung des Autors ausgeschlossen. Das Sicherheitsrisiko (Verlust des Satelliten) ist für die derzeitigen autonomen Planungssysteme nicht kalkulierbar und damit nicht vertretbar. Im Rahmen weiterführender Forschungsarbeiten muss das Sicherheitskonzept für autonom agierende Satelliten grundlegend überarbeitet werden. Das Ziel müsste darin bestehen, eine Methode zu entwickeln, das Sicherheitsrisiko für autonome Satelliten formal und quantitativ zu berechnen und mittels Beweisführung zu sichern.

Abschließend wäre es für die Entwicklung autonomer Planungssysteme generell hilfreich, einen Standard (z.B. Normen) zu definieren. Kapitel 4 gibt einen Überblick über den aktuellen Stand der Technik und der Forschung. Es haben sich bereits einige Architekturen und Begrifflichkeiten herausgebildet. Zur Vereinheitlichung würde der Autor daher die Aufnahme eines neuen Services für die autonome Planung in dem Packet Utilization Standard (PUS) der ECSS propagieren.

11 Anhang

- Protokolle und Graphiken der Messungen aus Abschnitt 8.2
- Dissertation in elektronischer Form (pdf)

12 Literaturverzeichnis

[1] H. Kayal, O. Balagurin und H. Wojtkowiak, „ASAP - A Sensor System for Autonomous Event Detection and on Board Planning," *Proc. of the Internation Astronautical Congress 2012,* 2012.

[2] H. Kayal, O. Balagurin und H. Wojtkowiak, „ASAP - Autonomer Bildsensor und Planungssystem für Nanosatelliten," *Tagungsband des Deutschen Luft- und Raumfahrtskongress 2012,* 2012.

[3] H. Wojtkowiak, O. Balagurin, G. Fellinger, B. Garcia und H. Kayal, „ASAP: Autonomous Dynamic Scheduling For Small Satellites," *Proc. 64th International Astronautical Congress, Beijing, China,* September 2013.

[4] H. Wojtkowiak, O. Balagurin, G. Fellinger, W. Fischer, B. Garcia und H. Kayal, „ASAP: Autonomous Onboard Mission Planning," *Proc. 65th International Astronautical Congress, Toronto, Canada,* September 2014.

[5] O. Balagurin, H. Kayal und H. Wojtkowiak, „Validation and Qualification of a CMOS based Miniature Star Tracker for Small Satellites," *Proc. The 4S Symposium 2012,* Juni 2012.

[6] S. Montenegro und J. Richardson, „RODOS operating system for Network Centric Core Avionics," *Research Gate,* 2008.

[7] T. Walter, A. Hilgarth, T. Mikschl und S. Montenegro, „VIDANA: A fault tolerant approach for a distributed data management system in nano-satellites," *Proc. 10th Symposium on Small Satellites for Earth Observation,* April 2013.

[8] F. Kempf, A. Hilgarth, A. Kheirkhah, T. Mikschl, T. Tzschichholz, S. Montenegro und K. Schilling, „Reliable Networked Distributed On-Board Data Handling Using a Modular Approach with Heterogeneous Components," *Proc. The 4S Symposium, Majorca, Spain,* May 2014.

[9] G. Fellinger, G. Dietrich, G. Fette, H. Kayal, F. Puppe, V. Schneider und H. Wojtkowiak, „ADIA - A Novel Onboard Failure Diagnostic System for Nanosatellites," *Proc. 64th International Astronautical Congress, Beijing, China,* September 2013.

[10] J. Eickhoff, Onboard Computers, Onboard Software and Satellite Operations, Berlin Heidelberg: Springer-Verlag, 2012.

[11] R. Sandau, International Study on Cost-Effective Earth Observation Missions, Berlin: A.A. Balkema Publishers, 2006.

[12] H. Kayal, „Universität Würzburg, Lehrstuhl für Informatik VIII," [Online]. Available: http://www8.informatik.uni-wuerzburg.de/mitarbeiter/kayal0/. [Zugriff am 1 8 2016].

[13] T. H. Cormen, C. E. Leiserson, R. Rivest und C. Stein, Algorithmen - Eine Einführung, München: Oldenbourg Wissenschaftsverlag GmbH, 2010.

[14] W. Truszkowski, L. Hallock, C. Rouff, J. Karlin, J. Rash, M. G. Hinchey und R. Sterritt, Autonomous and Autonomic Systems, Berlin: Springer, 2009.

[15] W. Ley und K. Wittmann, Handbuch der Raumfahrttechnik, 3. Auflage Hrsg., W. Hallmann, Hrsg.,

München: Carl Hanser Verlag, 2008.

[16] P. Fortescue, G. Swinerd und J. Stark, Spacecraft System Engineering, 4. Hrsg., Chichester: John Wiley & Sons Ltd., 2011.

[17] European Cooperation for Space Standardization, Hrsg., *ECSS-E-70-41A,* Noordwijk, The Netherlands: ESA-ESTEC, 2003.

[18] T. C. C. f. S. D. S. CCSDS, *CCSDS 130.2-G-2: Space Data Link Protocols - Summary of concept and rationale,* 2012.

[19] J. Teich und C. Haubelt, Digitale Hardware/Software-Systeme, Synthese und Optimierung, Springer-Verlag Berlin Heidelberg, 2007.

[20] P. Brucker, Scheduling Algorithms, Berlin Heidelberg: Springer-Verlag, 2007.

[21] P. Brucker und S. Knust, Complex Scheduling, Berlin Heidelberg: Springer-Verlag, 2012.

[22] M. L. Pinedo, Scheduling, Springer Science+Business Media, 2012.

[23] L. Delligatti, SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison Wesley, 2013.

[24] S. Friedenthal, A. Moore und R. Steiner, A Practical Guide to SysML, Waltham: Morgen Kaufmann, 2012.

[25] R. Sherwood, S. Chien, D. Tran, B. Cichy, R. Castano, A. Davies und G. Rabideau, „The EO-1 Autonomous Sciencecraft," *Proc. Small Satellite Conference,* 2007.

[26] G. Beaumet, „Continuous Planning for the Control of an Autonomous Agile Satellite," *Proc. International Conference on Automated Planning and Scheduling,* 2006.

[27] E. C. F. S. S. ECSS, *ECSS-E-ST-70-11C, Space engineering - Space segment operability,* 2008.

[28] A. S. Fukunaga, G. Rabideau, S. Chien und D. Yan, „ASPEN: A Framework for Automated Planning and Scheduling of Spacecraft Control and Operations," *Proc. International Symposium on AI, Robotics and Automation in Space,* 1997.

[29] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins und D. Tran, „ASPEN - Automated Planning and Scheduling for Space Mission Operations," *Proc. International Conference on Space Operations,* 2000.

[30] S. Chien, R. Knight, A. Stechert, R. Sherwood und G. Rabideau, „Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling," *Proc. 5th International Conference on Artificial Intelligence Planning and Scheduling,* 2000.

[31] G. Rabideau, R. Knight, S. Chien, A. Fukunaga und A. Govindjee, „Iterative repair planning for spacecraft operations using the ASPEN system," *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space,* 1999.

[32] S. Chien, R. Knight, A. Stechert, R. Sherwood und G. Rabideau, „Integrated planning and execution for autonomous spacecraft," *Proc. IEEE Aerospace Conference,* 1999.

[33] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. Hengemihle, J. D'Agostino, S. Shulman, S. Ungar, T. Brakke, D. Boyer, J. V. Gaasbeck, R. Greeley, T. Doggett, V. Baker, J. Dohm und F. Ip, „The EO-1 Autonomous Science Agent," *Proc. Third International Joint Conference on Autonomous Agents and Multiagent Systems,* 2004.

[34] S. Chien, R. Sherwood, G. Rabideau, R. Castano, A. Davies, M. Burl, R. Knight, T. Stough, J. Roden, P. Zetocha, R. Wainwright, P. Klupar, J. V. Gaasbeck, P. Cappelaere und D. Oswald, „The Techsat-21 Autonomous Space Science Agent," *Proc. First international joint conference on autonomous agents and multiagent systems,* 2002.

[35] S. Chien, „Integrated AI in space: the autonomous sciencecraft on earth observing one," *Proc. 21st national conference on artificial intelligence,* 2006.

[36] R. Sherwood, S. Chien, R. Castano und G. Rabideau, „Autonomous planning and scheduling on the TechSat 21 Mission," *Proc. Australian Joint Conference on Artificial Intelligence,* 2002.

[37] T. L. Mims und National Aeronautics and Space Administration, *Use of Spacecraft Command Language for Advanced Command and Control Applications,* National Aeronautics and Space Administration (NASA), 2008.

[38] M. Lemaitre und G. Verfaillie, „Interaction between reactive and deliberative tasks for on-line decision-making," *Proc. Workshop on "Planning and Plan Execution for Realworld Systems",* 2007.

[39] G. Beaumet, G. Verfaillie und M.-C. Charmeau, „Autonomous Planning for an Agile Earth-Observing Satellite," *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space,* 2008.

[40] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver und N. Bataille, „Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology, Volume 6, Issue 5,* pp. S.367-381, 2002.

[41] M. Ricard und S. Kolitz, „The ADEPT Framework for Intelligent Autonomy," *Proc. Intelligent Systems for Aeronautics,* 2003.

[42] M. Abramson, D. Carter und S. Kolitz, „The Design and Implementation of DRAPER´s Earth Phenomena Observing System (EPOS)," *Proc. AIAA Space Conference,* 2001.

[43] M. Ricard, M. Abramson, D. Carter und S. Kolitz, „The Earth Phenomena Observing System: Intelligent Autonomy for Satellite Operations," *Proc. Intelligent Systems for Aeronautics,* 2003.

[44] G. Verfaillie und M.-C. Charmeau, „A generic modular architecture for the control of an autonomous spacecraft," *Proc. International Workshop on Planning and Scheduling in Space,* 2006.

[45] M.-C. Charmeau und E. Bensana, „AGATA, A Lab Bench Project for Spacecraft Autonomy," *Proc. International Symposium on Artifical Intelligence, Robotics and Automation in Space,* 2005.

[46] R. Steel, M. Niézette, A. Cesta, S. Fratini, A. Oddi, G. Cortellessa, R. Rasconi, G. Verfaillie, C. Pralet, M. Lavagna, A. Brambilla, F. Castellini, A. Donati und N. Policella, „Advanced Planning and Scheduling Initiative - MrSpock Aims for XMas," *Proc. Workshop on Artificial Intelligence in Space,* September 2009.

[47] B. Wille, M. T. Wörle und C. Lenzen, „VAMOS - Verification of Autonomous Mission Planning On-Board a Spacecraft," *Proc. 19th IFAC Symposium on Automatic Control in Aerospace,* 2013.

[48] M. T. Wörle und C. Lenzen, „Ground Assisted Onboard Planning Autonomy with VAMOS," *Proc. International Workshop for Planning and Scheduling in Space,* 2013.

[49] C. Lenzen, M. T. Wörle, T. Göttfert, F. Mrowka und M. Wickler, „Onboard Planning and Scheduling Autonomy within the Scope of the FireBird Mission," *Proc. SpaceOps 2014 - 13th International Conference on Space Operations,* 2014.

[50] A. R. Carrel und P. L. Palmer, „An evolutionary algorithm for near-poptimal autonomous ressource management," *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space,* 2005.

[51] A. Carrel und P. Palmer, „Adaptive Resource Modelling for Autonomous Planning and Scheduling," *Proc. 5th International Workshop on Planning and Scheduling in Space,* 2006.

[52] C. Ebert, Systematisches Requirements Engineering - Anforderungen ermitteln, dokumentieren, analysieren und verwalten, Heidelberg: dpunkt.verlag GmbH, 2014.

[53] P.-Q. Pan, Linear Programming Computation, Berlin Heidelberg: Springer-Verlag, 2014.

Eidesstattliche Versicherung und Erklärung

Vor- und Zuname: Harald Wojtkowiak
Geburtsdatum: 01.04.1982
Geburtsort: Ludwigshafen am Rhein

Hiermit versichere ich an Eides statt,

- dass ich die vorgelegte Dissertation selbstständig und ohne unzulässige fremde Hilfe, d.h. insbesondere ohne Hilfe einer kommerziellen Promotionsberatung, angefertigt und verfasst habe, dass alle Hilfsmittel und sonstige Hilfen angegeben und dass alle Stellen, die ich wörtlich oder dem Sinne nach aus anderen Veröffentlichungen entnommen habe, kenntlich gemacht worden sind;
- dass die Dissertation in der vorgelegten oder einer ähnlichen Fassung noch nicht zu einem früheren Zeitpunkt an der Julius-Maximilians-Universität Würzburg oder einer anderen in- oder ausländischen Hochschule als Dissertation eingereicht worden ist;
- dass ich die Gelegenheit zur Promotion nicht kommerziell vermittelt bekam und insbesondere nicht eine Person oder Organisation eingeschaltet habe, die gegen Entgelt Betreuer bzw. Betreuerin für die Anfertigung von Dissertationen sucht.

Hiermit erkläre ich,

- dass ich die Regeln der Universität Würzburg über gute wissenschaftliche Praxis eingehalten habe.

Unterschrift Ort, Datum