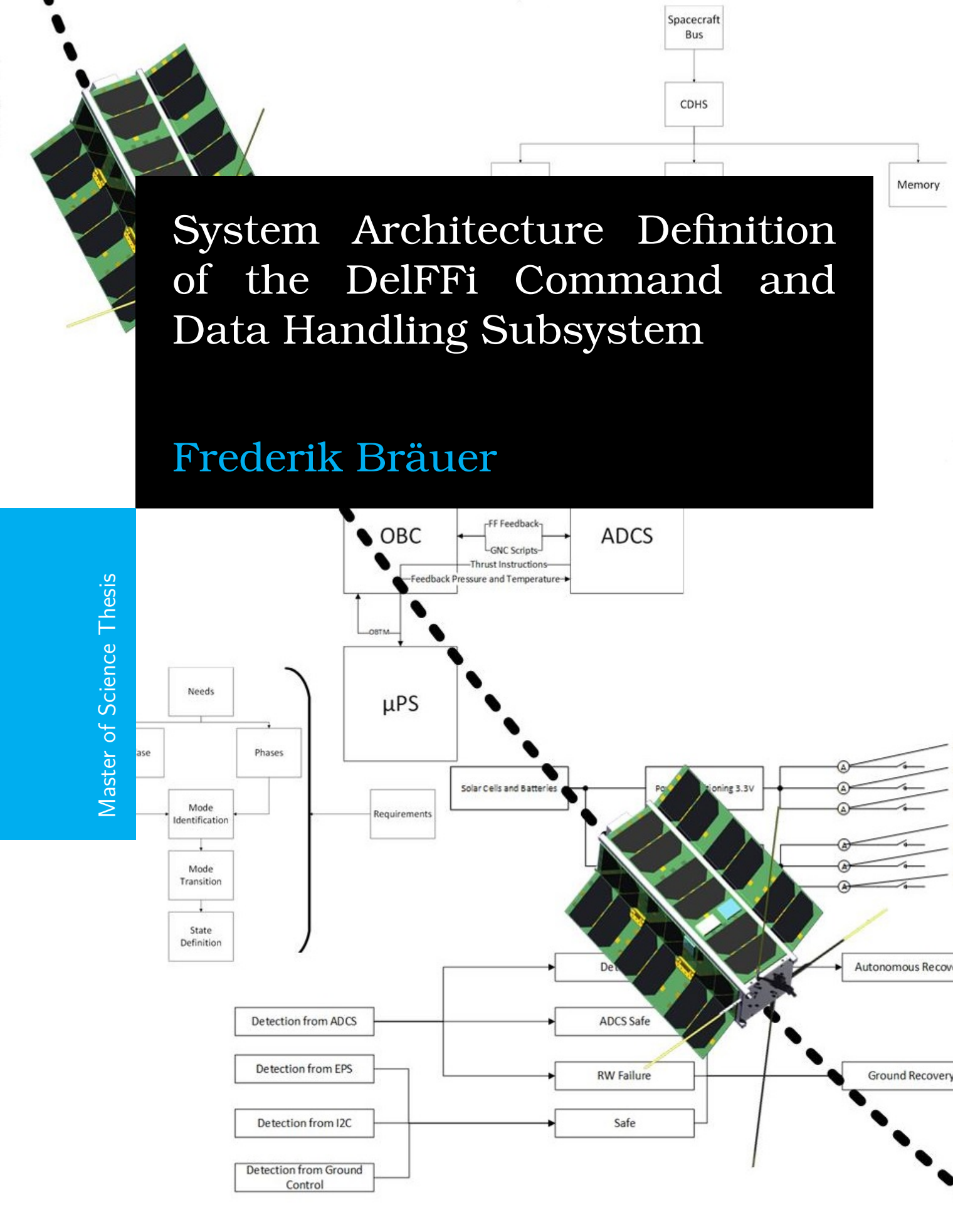
System Architecture Definition of the DelFFi Command and Data Handling Subsystem

Frederik Bräuer

Master of Science Thesis



TU Delft Delft University of Technology

Space System Engineering

System Architecture Definition of the DelFFi Command and Data Handling Subsystem

Master of Science Thesis

For the degree of Master of Science in Space Engineering at Delft University of Technology

Frederik Bräuer

22.06.2015

Faculty of Aerospace Engineering(AE) · Delft University of Technology

Abstract

DelFFi is a CubeSat mission with the objective to demonstrate formation flying and explore the composition of the upper atmosphere. The research performed in this master thesis focusses on the development of the system architecture of the Command and Data Handling Subsystem of the two identical satellites. Specifically three questions are answered: What is the current status and what work is remaining? How can system modes and states be defined for the system architecture of a CubeSat? Finally, what are the possibilities to include error detection and handling on a nanosatellite.

The Command and Data Handling Subsystem consists of two general components: Hardware and Software. Analysis shows that the hardware components are close to a mature Technical Readiness Level (TRL) between four and nine. Conversely, the development of the on-board software is at an early stage of development with a TRL for most items equal to, or below, three.

As a consequence of the second research question, this work attempts to determine what the system is performing at certain moments in time by defining system phases, modes and states. On the basis of academic and industry literature, a 6-step system engineering methodology has been developed in this thesis for the application to CubeSat missions. Beginning from the mission objectives, the operational stakeholders and their uses cases, a top-down flow allows the definition of the system modes and corresponding states. Implemented for DelFFi, a total 10 modes and 33 states have been defined.

One special mode is the safe mode, which assures the survival of the satellite in non-nominal conditions. The spacecraft transition into this mode by ground command or autonomously due to Failure, Detection, Isolation and Recovery procedures(FDIR). On the basis of analysis into the different CDHS configuration items FDIR procedures have been implemented successfully into the DelFFi software architecture.

To summarize the research performed within this thesis successfully furthered the system architecture of the DelFFi CDHS by defining the System Modes and Phases and implementing FDIR procedures.

Table of Contents

List of Figures

List of Tables

Preface

About 10 months ago, in a place not too far away, I completed my internship in the Operations and System Engineering of BepiColombo. The time in the project showed me that for my career in space, I want to work in the area of Operations Engineering.

Returning to Delft, I decided to write my literature study about the Command and Data Handling Subsystem of a CubeSat mission named DelFFi. The topic was exciting as the CDHS controls and operates a spacecraft and can be considered to be the electronic brain of satellite. Following an 8 week study of the different technologies and concepts used, the next step was to continue the development of the system architecture for DelFFi. This document summarizes the work and research performed from mid of November 2014 to June 2015.

A section focusses on the concept of operations for the CDHS, allowing me to gain further experience in the operational planning of a space mission. Within my research I had the opportunity to define system modes, phases and states of the mission. The second part focusses how failures can be handled and gave insight into the software and hardware level interactions of the CDHS.

The work and research performed allowed me to gain understanding of an important field in the development of spacecraft, which is not directly taught as part of the Space Engineering master. With the experiences from a large interplanetary probe and a challenging CubeSat mission I feel prepared to work on future space missions.

Acknowledgements

I would like to thank my supervisor Ir. Jasper Bouwmeester for his assistance during the research and writing of this thesis. Additionally I want to thank Nuno dos Santos for his input and bringing the world of electrical engineering closer to an aerospace engineer. Furthermore I would like to thank the whole DelFFi team for challenging discussions and input. For the future I wish everyone involved the best for the further development of the mission.

During my time in Delft I had the pleasure to get to know many different people from around the world. Especially I would like to thank Svenja Woicke, Nora Kind, Ingo Gerth and Bjöern Roscher. My time in EUROAVIA showed me the different countries of Europe from Portugal to Croatia. I would like to thank exemplary Jacqueline Chindea and Marco Marino. In the past 7 years Delft and the Netherlands became my home and I hope to return in the future.

My internship at Airbus Defence and Space showed me the world of large spacecrafts for the exploration of our Solar System. I would like to thank Susanne Fugger for guidance and input during my time in the BepiColombo project.

Finally I would like to thank my Family for supporting me to go to the Netherlands to become a rocket scientist

Delft, University of Technology — Frederik Bräuer
22.06.2015

Glossary

List of Acronyms

ADCS	Attitude Determination and Control System
AHP	Analytical Hierarchy Process
CCSDS	Consultive Committee for Space Data Systems
CDHS	Command and Data Handling System
COTS	Commercial off the Shelf
CRC	Cyclic Redudancy Check
CTA	Configuration Tree Analysis
DelFFi	Delft Formation Flying Experiment
DRB	Daughter Board
DSSB	Delft Standard System Bus
ECSS	European Cooperation for Space Standardization
EM	Engineering Model
EPS	Electronic Power System
ESA	European Space Agency
FDIR	Failure, Detection, Isolation, Recovery
FF	Formation Flying
FIPEX	Flux-(Phi)-Probe-Experiment
FM	Flight Model
FMEA	Failure Mode and Effect Analysis

FTA	Fault-Tree Analysis
ISIS	Innovation Solutions in Space
I2C	Inter-Integrated Circuit
ICD	Interface Control Document
ITU	International Telecommunication Union
LEO	Low Earth Orbit
LEOP	Launch and Early Operation Phase
LOS	Loss of Signal
LOM	Loss of Mission
MCU	Main Computing Unit / Microcontroller
NACK	Not Acknowledged
NASA	National Aeronautics and Space Administration
OBC	On-board Computer
OBSW	On-Board Software
OBTM	On-Board Telemetry
PCB	Printed Circuit Board
QB50	Atmospheric Research Network
RAM	Random Access Memory
RTOS	Real-Time Operating System
RW	Reaction Wheels
S/C	Spacecraft
SCL	Serial Clock Line
SDA	Serial Data Line
SE	Systems Engineering
SEE	Single Event Effect
SEU	Single Event Upset
SEL	Single Event Latchup
SPI	Serial Peripheral Interface
SU	Science Unit

TC	Telecommand
TID	Total Ionization Dose
TLE	Two-Line Element
TM	Telemetry
TRL	Technical Readiness Level
TU Delft	Delft University of Technology
UART	Universal Asynchronous Receiver/Transmitter
UML	Unified Modeling Language
VKI	Van Karman Institute

List of Symbols

β	Ballistic Coefficient
ΔV	Velocity differential
A	Surface Area in Flow-direction
C_d	Drag Coefficient
I_{switch}	Current EPS Channel Output
M	Satellite Mass
P_{cons}	Power consumed by S/C
S	Body Surface Areas
s_{sep}	Spacecraft separation distance
$T_{battery}$	Battery Temperature
v_{diff}	Velocity differential in along track direction
[deg]	Degree
$[\frac{\text{deg}}{s}]$	Degree per Second
$[\frac{kg}{m^2}]$	Kilogram per square meter
$[\frac{m}{s}]$	Meter per second
$[GB]$	Gigabyte
$[m^2]$	Square meter
[KB]	Kilobyte
[km]	Kilometer
[kRad]	Absorbed Radiation Dose
[Mhz]	Megaherz
[min]	Minutes
[s]	Seconds
[W]	Watts

Chapter 1

Introduction

Following from the heritage of Delfi-C3 and Delfi-n3Xt, DelFFi is the third CubeSat project developed at the Faculty of Aerospace Engineering in Delft. The DelFFi mission has the objective to give students hands-on experience, perform measurements of the upper atmosphere within the QB50 network and demonstrate formation flying with two identical CubeSats: Delta and Phi. A general discussion about the DelFFi mission can be found in chapter 2.

The research objective for the master thesis is to further develop the CDHS system architecture of a CubeSat formation flying mission, by developing a top-down methodology for the determination of the system modes and implementation of a Failure, Detection, Isolation and Recovery procedures within the constraints of a CubeSat.

This thesis focuses on the advancement of the system architecture of the Command and Data Handling System (CDHS). The research objective is to further develop the CDHS system architecture of a CubeSat formation flying mission, by developing a top-down methodology for the determination of the system modes and implementation of a Failure, Detection, Isolation and Recovery procedures within the constraints of a CubeSat. The research objective and expanded research question are detailed in section 1-2 and the evolution of the project in section 1-3.

The CDHS, which can be considered to be the electronic brain of the satellite, allows for autonomous operations of the system. Prior work by Remco Schoemaker and Gang Liu defined the format of the telemetry and began the development of the software architecture. The On-board Computer (OBC) has been developed by Nuno dos Santos and is currently undergoing initial testing. The current status and a general introduction into the different CDHS components of DelFFi is discussed in chapter 3.

At the beginning of the research, it became apparent that within the DelFFi project the terms system modes and states are not well defined. Consequently, the need and motivation arose to derive the system modes and states for the DelFFi mission. Research into definitions of system modes, in academic literature and engineering standards, showed that the process of defining modes and states from the mission objective of a Spacecraft (S/C), is currently not well defined for CubeSats. Chapter 4 develops, on the basis of literature, a top-down

methodology to determine the system modes and states for a CubeSats. From the DelFFi mission objectives 10 system modes and 33 supporting states are implemented in the the system architecture of the CDHS.

A part of the research objective is to answer how to implement Failure, Detection, Isolation, Recovery (FDIR) procedures into a CubeSat, such as DelFFi. A part of failure mitigation is to include FDIR procedures into the software routines of the S/C. Implementation of FDIR procedures are important as the subsystem forms the interface between the different subsystems of the S/C and the operations team on-ground. With this function in mind, failure events are identified and analysed on the bases of the configuration items. The definition and implementation of FDIR procedures is discussed in chapter 5.

On the basis of the discussion in chapter 3, 4 and 5, conclusions are provided in chapter 6 and recommendations for the future work within DelFFi and possible future Delfi mission are provided in chapter 7.

1-1 Research Motivation

The idea to continue the development of the CDHS subsystem was first discussed in August 2014 as a follow up to the work performed by Remco and Gang. At the time the development and testing part of the DelFFi project was planned to be finished in September 2015. Thus from the project perspective there was the need to move the development of the CDHS in software and hardware from conception to implementation and testing. The original work packages included support in the programming of the OBC and preparing and executing tests with the Engineering Model (EM) and Flight Model (FM). For various reasons, of which some are detailed in section1-3, the project had to evolve.

The need to define system modes arose as part of the work in the definition of the on-board software. After many discussions, it became apparent that definition of these modes needs to be based on a methodology and move away from an arbitrary selection that varied between the different subsystems. This originally small part of the project grew to be a major part of the work performed. In the discussions about the system modes, a safe mode for the system was defined. This escalated into an investigation of how to enter the safe mode and how to react to failure events on the CDHS. Both discussions were required for further developing the system architecture and will form the backbone for the future work on the subsystem.

1-2 Research Objective and Questions

Within this section the formal research objective and research questions is presented and the questions which will be answered in the subsequent chapters are derived. For defining the objective first the context and scope needs to be established.

Context for the MSc thesis can be can be described from two perspectives. One is the educational place in the MSc curriculum at the TU Delft. The thesis is located at the end of the master degree, in which the student is expected to combine and expand knowledge in a specific field and perform research to contribute to the standing body of knowledge. The thesis work itself is also within the context of the DelFFi project.

Scope is defined by the time and topic constraints. A satellite project such as DelFFi is a complex project which requires expertise in different fields, from mission planning to the design of the CDHS. While compared to the larger satellites the complexity is limited, dedicated engineers for the different subsystems are still needed. In combination with time constraint, this means a single engineer or MSc student can only work on a limited set of work packages. For this thesis, the work packages would be to continue the definition of the system architecture by developing the system modes and the FDIR procedures. The time constraint is defined by the 42 assigned ECTS, which correspond to approximately 7 months of full-time work.

Hence from context and scope the research objective can be derived:

The research objective for the master thesis is to further develop the CDHS system architecture of a CubeSat formation flying mission, by developing a top-down methodology for the determination of the system modes and implementation of a Failure, Detection, Isolation and Recovery procedures within the constraints of a CubeSat.

From the objective the research questions to be answered are listed below. The first question complex is mainly answered in chapter 3, the second in chapter 4 and the third in chapter 5. Finally, general conclusions and recommendation on the basis of these questions are provided in chapter 6 and 7.

1. Command and Data Handling System of DelFFi
   (a) What is the current development status of DelFFi?
   (b) What improvements can be identified for future Delfi missions?
2. System Phases, Modes and States
   (a) How are States and Modes defined in academia and industry?
   (b) What general approach can be used to develop Modes and States from the objectives of a CubeSat mission?
   (c) What are the system phases, modes and states for DelFFi?
3. Failure, Detection, Isolation, Recovery for DelFFi
   (a) How are FDIR procedures implemented on traditional spacecraft?
   (b) Which failure analysis methodology can be used to identify failure events of the DelFFi CDHS?
   (c) How can a FDIR procedure be implemented for DelFFi?

1-3 Project Planning and Evolution

The thesis proposal developed as part of [1] differs from the research performed within this document. There are several reasons which required the evolution of the project. In the beginning of the project, it was expected that a dedicated embedded systems programmer would be available for the implementation of the software architecture.

Furthermore, it has been expected that most hardware, or subsystems, would be available in March 2015. This resulted in the original plan to perform testing between the CDHS and its interfaces. At the end of the project only the Flux-(Phi)-Probe-Experiment (FIPEX), prototypes of the OBC and, Daughter Board (DRB) are available.

As a consequence, the focus on the development of the software architecture moved from implementation and testing of the DelFFi satellites to a system engineering discussion of states and modes and a concept for the implementation of FDIR procedures. Figure 1-1 shows the activity flow of the project. The performed research started to deviate from the original planning in January after the baseline definition. At the time it was assumed that in February a programmer would be available for implementing of the On-Board Software (OBSW), and a baseline had to be established that would allow testing of the CDHS engineering model. However this was not possible which forced the project from practical implementation to a more system engineering focused discussion. The need for the development of a methodology for system modes arose and became the major part of the discussion within this document. A common item in the original planning and the evolved project is the discussion on the FDIR procedures. In figure 1-1 the general flow of the project is presented, with the originally planned items shaded in grey.

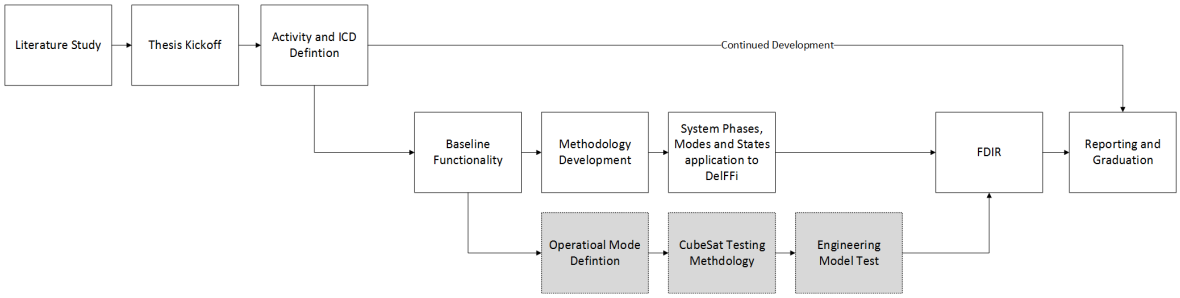


Figure 1-1: Project Timeline

1-4 Thesis Outline

This thesis presents the current status of the DelFFi Command and Data Handling System with a specific focus on the development of the system modes, phases and states and the implementation of FDIR procedures. Chapter 2 provides a summary of the DelFFi mission, followed by 3 which provides background on the components of the CDHS and summarizes the current development status. In chapter 4 a methodology is developed to determine the system modes and states of a CubeSats. Continuing is chapter 5, which discusses the implementation of FDIR procedures. Chapter 6 provides the conclusions to the research, followed by the recommendations in chapter 7.

Chapter 2

DelFFi Mission

QB50 is an international project under the leadership of Van Karman Institute (VKI) to research composition of the upper atmosphere. The plan is to launch around 50 CubeSats in 2016/2017 which allow ,after dispersion, to map the atmosphere over the whole globe providing unprecedented temporal resolution. Delft University of Technology (TU Delft) is contributing to the mission with two satellites, Delta and Phi, with the mission statement:

The DelFFi mission shall demonstrate autonomous formation flying and provide enhanced scientific return within QB50 from 2015 onwards, by utilizing two identical triple unit CubeSats of TU Delft which further advance the Delfi n3Xt platform. [2]

The functions and specifications of a spacecraft are defined by its requirements, which are documented for DelFFi in [3].

2-1 Objective

From the mission statement two objectives can be identified: Formation Flying using two CubeSats and the providing scientific return in the scope of the Atmospheric Research Network (QB50) mission. Within this section first the QB50 science objective is discussed in section 2-1-1, followed by a discussion on the formation flying objective in section 2-1-2

2-1-1 QB50

The scientific objective of QB50 is to explore the upper atmosphere between 200-380 [km], which currently can only be measured using sounding rockets. 50 CubeSats will be used to map the upper atmosphere, each equipped with one of two instruments.[4] For Delft Formation Flying Experiment (DelFFi) the Flux-(Phi)-Probe-Experiment (FIPEX) instrument, developed by Technische Universitẗ Dresden, has been to chosen to be carried on Delta and Phi. The instrument has the capability to measure the molecular gases in the upper atmosphere, with the working principles published in [5].

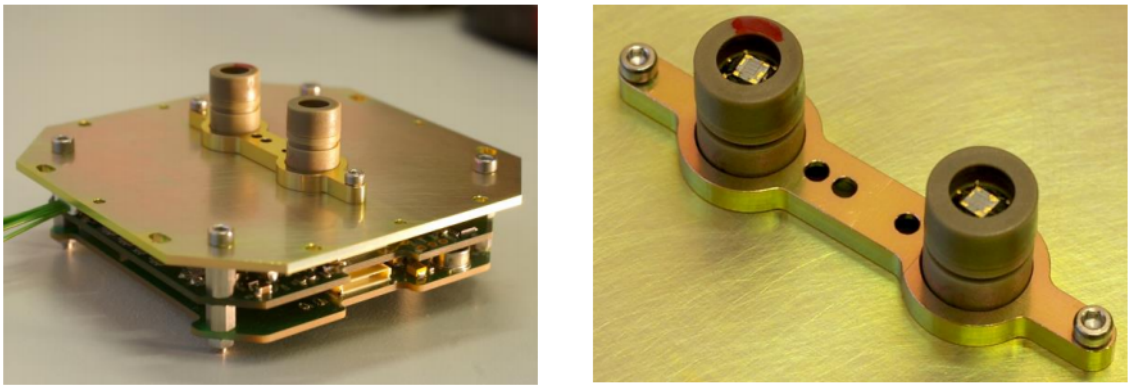


Figure 2-1: FIPEX Payload (from [6])

2-1-2 Formation Flying

Following the development of Delfi C3 and Delfi n3Xt the DelFFi satellites have the objective to demonstrate formation flying capabilities with CubeSats. The objective requires the development of formation flying concepts and a propulsion system.

Concept

An initial study about using formation flying has been performed by [7] and further refined in [8]. The initial study describes the basis for the mission objectives. Both satellites after being released from the Spacecraft (S/C) will begin to drift apart due to an initial velocity and drag differential. For keeping a formation both DelFFi satellites require propulsion capabilities to maintain a separation distance of 1000 $[km]$. The initial study describes the control of a formation using relative inclination and eccentricity vector control. Different scenarios have been developed for the formation flying operational concept, which are shown in figure 2-1. Currently the implementation of formation flying is developed and will be published in the future in [9].

Propulsion

A integral part of formation flying is the capability to change the orbital elements of both Delta and Phi. For DelFFi a MEMS resistor-jet thruster is developed on the research of [10], of which the current design is shown in figure 2-3.

2-2 Overview of Subsystems

A general overview of one of the DelFFi satellites is shown in figure 2-4. For power generation two solar array wings are mounted to the triple unit CubeSat structure which transfer power to the Electronic Power System (EPS) and battery board. This subsystem is a Commercial off the Shelf (COTS) product from GOMSpace. The payloads for the mission are the FIPEX,

# and Name Scenario	Guidance	Navigation	Control	Required
1. Basic	On-ground	TLE uploaded to 1 s/c	Single thruster	1 s/c, onground guidance, U/L to 1 s/c
2. Onboard	Onboard	TLE uploaded to 1 s/c	Single thruster	1 s/c, onboard guidance, U/L to 1 s/c
3. Distributed	On-ground	TLE uploaded to 2 s/c	Single, dual thrusters	2 s/c, onground guidance, U/L to 2 s/c
4. Distributed onboard	Onboard	TLE uploaded to 2 s/c	Single, dual thrusters	2 s/c, on-board guidance, U/L to 2 s/c
5. Distributed coordinated	On-board	Relative TLE onboard and exchanged via ground	Single, dual thrusters	2 s/c, on-board guidance, U/L and D/L for 2 s/c
6. Onboard ISL-based relay	On-board	Relative TLE computed onboard, relayed via ISL	Single, dual thrusters	2 s/c, on-board guidance, U/L and D/L for 1 s/c
7. Distributed coordinated ISL-based	On-board	Relative TLE computed onboard and exchanged with ISL	Single, dual thrusters	2 s/c, on-board guidance, U/L and D/L for 2 s/c
8. Full autonomous formation flying	On-board	Determined onboard by relative navigation RF payload	Single, dual thrusters	2 s/c, on-board guidance, relative navigation RF payload

Figure 2-2: Formation Flying Concepts(from [8])

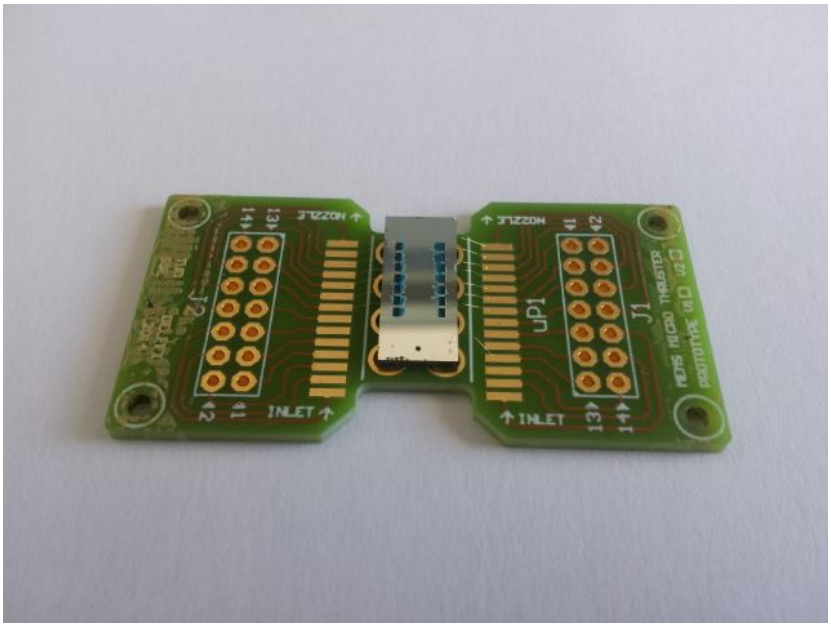


Figure 2-3: Thruster Prototype

the propulsion systen and GAMALink. Communication to ground is achieved by the TRXUV transceiver and AntS antennas acquired from Innovation Solutions in Space (ISIS). For achieving the mission objectives the attitude of the S/C needs to be stablized which is achieved by the in-house developed Attitude Determination and Control System (ADCS). The Command and Data Handling System (CDHS) consists of a non-redundant On-board Computer (OBC), the Daughter Board (DRB), the memory storage and the data buses. A more extensive discussion on the Command and Data Handling System is provided in chapter 3 of this document.

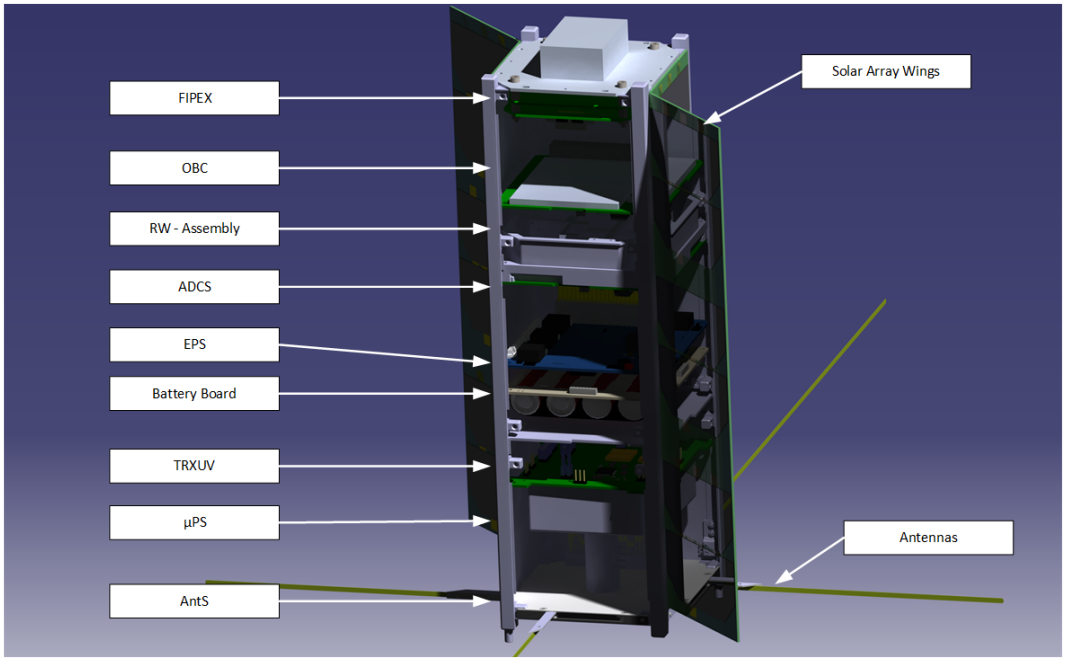


Figure 2-4: DelFFi Subsystem Overview

Chapter 3

Command and Data Handling of DelFFi Overview

Within this chapter an overview of the current state of the Delft Formation Flying Experiment (DelFFi) Command and Data Handling System (CDHS) is provided. In the beginning, prior work is detailed in section 3-1, followed by a general overview of the subsystem and its components in section 3-2. Section 3-3 introduces the current software architecture. Finally the open work packages for development of the subsystem in general are detailed in 3-3. Parts of the discussion have been adapted from prior work of the author in [1].

The CDHS can be divided into two components: hardware and software. Hardware is considered to be every physical component of the systems including, but not limited to, the On-board Computer (OBC), peripheral Memory and interfaces to other subsystems. The software components consists of the On-Board Software (OBSW) and the supporting drivers.

3-1 Prior Work

The definition of the system architecture of the DelFFi CDHS is performed on the basis of heritage from Delfi-n3xt and C3 and work performed by prior students and staff members.

Flight Heritage

The experience from Delfi-n3xt and Delfi-C3 form the basis for the development of the CDHS. The non-redundant Main Computing Unit / Microcontroller (MCU) for DelFFi has been selected on the basis flight-heritage from both predecessors and the required use of Commercial off the Shelf (COTS) components.[11] More information is provided in section 5-3-1. Similarly the Inter-Integrated Circuit (I2C) bus has been selected for available experience within the team and available COTS components. A detailed discussion on the lessons learnt from Delfi-C3 and Delfi-n3xt is provided in 5-2-3.

Prior students and staff

The development of a space system is a project which needs the expertise of different fields of engineering. For the discussion within this document research performed in [12] has been used, which contains the lessons learnt from Delfi-n3xt, a requirement analysis for the DelFFi CDHS and the definition of the telemetry packages.

The software architecture on the basis of Delfi-n3xt has been defined by a former staff member. Research performed within this discussion has been implemented into the system architecture and new versions of the activity flows and Interface Control Document (ICD) have been released to the team.

3-2 Subsystem Overview

An overview of the hardware part or physical layer of the CDHS is shown in figure 3-1. As shown in the figure the hardware components of the CDHS are the OBC, the Daughter Board (DRB) the peripheral memory and the data interfaces to the different subsystems. The software component of the CDHS is explained in section 3-3.

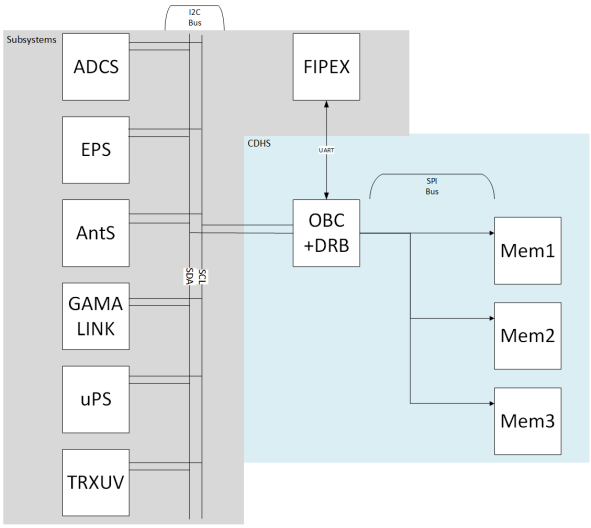


Figure 3-1: DelFFi CDHS Hardware Overview

3-2-1 MSP430F2418

The On-board Computer used on DelFFi is the MSP430F2418 built by Texas Instruments and is housed on a custom Printed Circuit Board (PCB). Detailed specification of the MCU can be found in the corresponding data sheet [13]. Summarizing the processor is a 16 bit architecture with a frequency of 16 [Mhz] and 116 [KB] Flash and 8[KB] RAM. A sketch of the OBC is shown in 3-2.

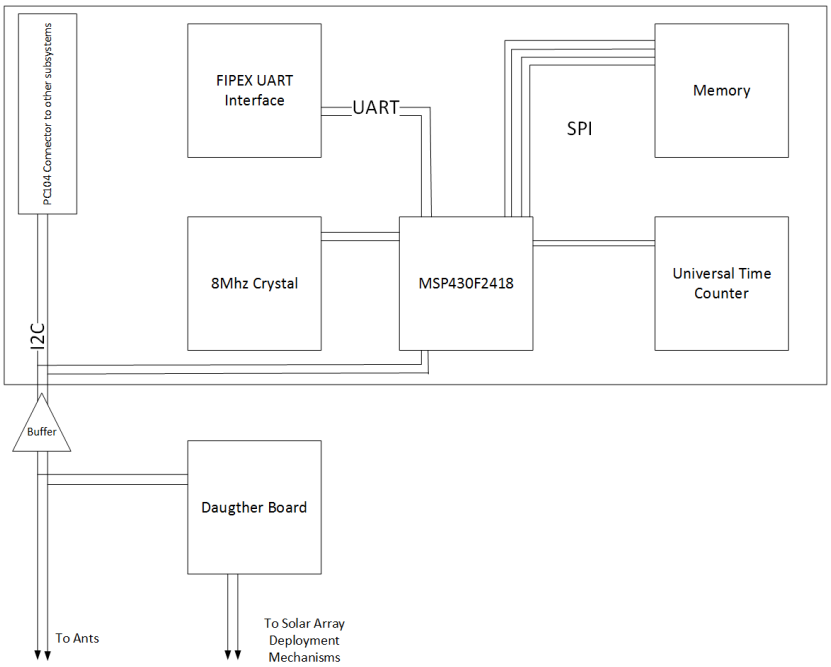


Figure 3-2: OBC PCB Overview

3-2-2 Memory

The peripheral memory, developed within [14], is required to store results of the Science Unit (SU) and for history playback of the Telemetry (TM). An image of the hardware component is shown in figure 3-3.

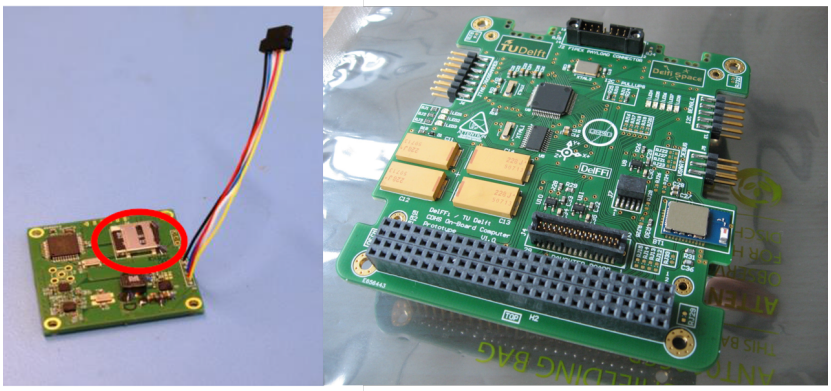


Figure 3-3: DelFFi Memory and OBC

3-2-3 Daughter Board

The daughter board is a dedicated MCU which has the objective to control the deployment of the appendages.The current prototype is shown in figure 3-4.

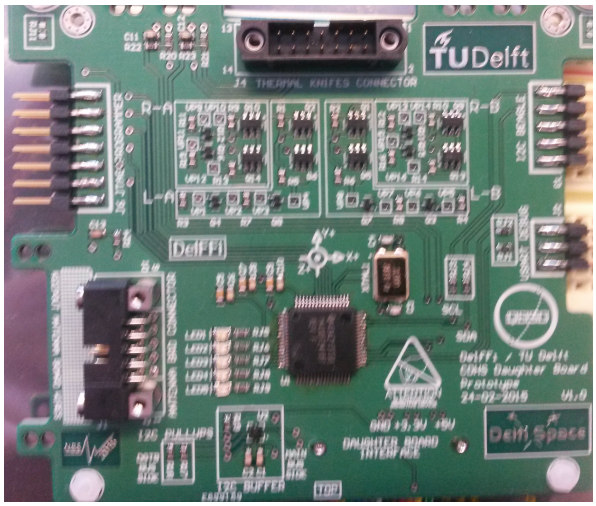


Figure 3-4: DRB Prototype

3-2-4 UART

On DelFFi the SU Flux-(Phi)-Probe-Experiment (FIPEX) requires to be connected to the CDHS via a Universal Asynchronous Receiver/Transmitter (UART) interface.[6] UART is the hardware part which translate from serial to parallel and vice-versa.

Connections with UART require one wire for the transmission of data. Data is transferred by pulling down the line to send data. A transmission initiates with a start bit, which is followed by 7 data bits. For the communication between CDHS and FIPEX no parity bit is required. [6] The transmission is ended using a stop bit. This means of the 10 bits transferred in a transmission 7 bits contain data. After the stop bit a new transfer can directly be initiated.

3-2-5 I2C

The I2C data interface has proven flight heritage within the Delfi project and many more CubeSat missions. Originally been developed by Phillips, it consists for DelFFi of a Master, the OBC, and slaves, every other subsystem except the FIPEX. The connection between the master and slaves is established using a Serial Clock Line (SCL) and a Serial Data Line (SDA). On previous Delfi satellites, the Delft Standard System Bus (DSSB) has been implemented to mitigate some of the known problems of I2C bus.(See 5-5) For easier implementation of COTS components the DSSB has not been implemented. A schematic of a I2C single master system is shown in figure 3-5.

Data Transfer principles

I2C requires all slaves to have a unique address on the bus. An address consists of 7 bits, which allows to connect a maximum of 112 devices to the I2C bus with some of the addresses reserved.

Two kind of data operations are possible using I2C: read and write operations. In both cases the transfer begins with the start condition and ends with a stop condition issued by the

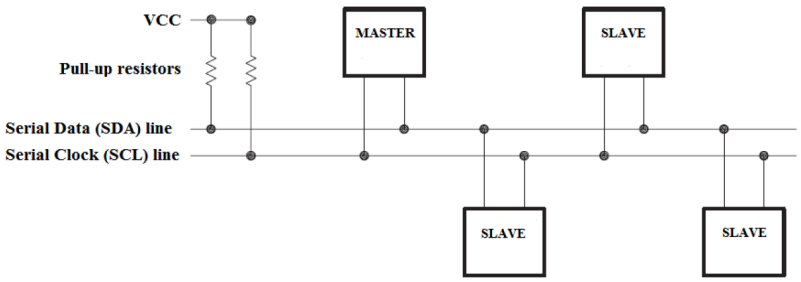


Figure 3-5: I2C setup (from [15])

master device. The start condition is initiated by the master pulling down the SDA line while the SCL is high and ends if the SDA is released while the SCL is high as well. This is shown in figure 3-6. After the master device issues a start condition all slaves listens for the address. The following bit indicates if the master wants to read, logical 1, or write, logical 0 from the slave device. The addressed slave responds by sending an acknowledge (ACK) bit by pulling down the SDA.

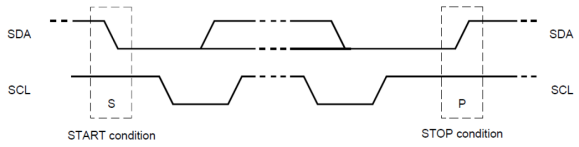


Figure 3-6: I2C Start and Stop Condition (from [15])

For writing the master sends a data byte (8 bit), after each byte the slave has to send an ACK bit. This is continued until all data is transferred and the master sends the stop bit. Such a transfer is shown in figure 3-7.

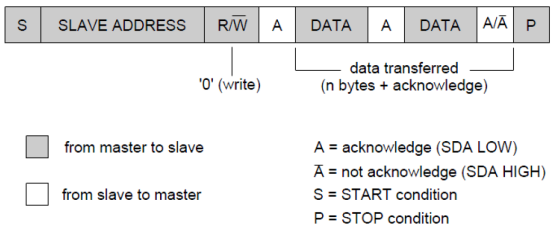


Figure 3-7: I2C Write (from [15])

The reading process is similar to the writing process. After the master address the slave it will sent a logic 1 to indicate it wants to read from the slave. The slave will then sent an acknowledgement bit and begin sending data to the master. The master will confirm the reception of the data byte with an ACK. If the data transfer is complete instead of an ACK the master will sent the stop condition. The process is shown in figure 3-7.

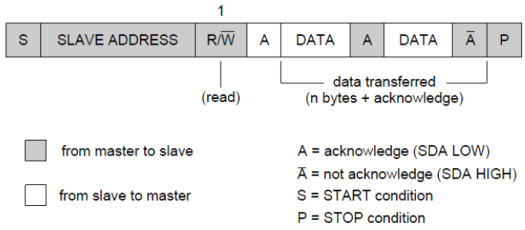


Figure 3-8: I2C Read (from [15])

I2C Addressing

The first part of a data transmission is the address of the slave device. In the I2C protocol 2 addressing schemes can be used depending on the amount of devices connected.

1. 7 bit
2. 10 bit

The 7-bit scheme allows to connect up to 127 devices and with 10 bit 1024 devices are possible. Not all addresses can be used, as some are reserved for special purposes. In table 3-1 all reserved addresses are shown. As a consequence the 7 bit addressing allows for a total of 112 different addresses. Currently for DelFFi 10 devices are connected using the I2C interface: ADCS, OBC, EPS, 2 addresses for TRXUV, uPS, GAMALINK, 2 addresses for the AntS and 1 for the DRB.

Table 3-1: I2C reserved addresses (adapted from [16])

10 Bit Addresses	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS Addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	Reserved for future purposes
11111XX X	Reserved for future purposes

Of the 112 possible addresses, 10 have to be selected for the DelFFi subsystems. The possibilities to assign the I2C addresses are as follows:

1. Sequentially
2. Randomly
3. Bit-Spaced

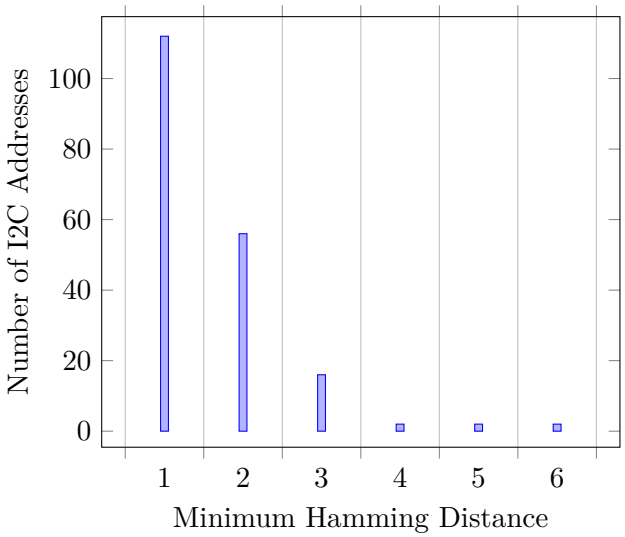


Figure 3-9: Number of I2C Addresses to Minimum hamming distance

Addresses, as shown in the table, are expressed in binary. It is important to be able to differentiate between addresses during testing and to reduce possible errors due to bit-flips caused by radiation events. This means an address written in binary should have at least a 2 bit different to all other addresses on a data bus. This assures that a single bit-flip, caused by a radiation events (see 5-2-2), cannot change an address to a other defined address.

From a mathematics perspective bit-spacing can be described by the hamming distance. This distance is, if the binary number is used as coordinates for the corner of a n-dimensional cube, the minimum required amount of edges of a path connecting two vertices. The method was first published by [17]. However for small binary number a computer can be used to compare the numbers position by position using XOR logic.

Comparing bit spacing to the sequentially and randomly it has the advantage of assuring that each selected address is clearly identifiable during testing and reduces the possible impact of bit flips. The two major disadvantages of bit-spacing is that the amount of addresses is limited and that COTS components can require specific addresses, hence further reducing the list of possible addresses. A shared disadvantage with random selection is that logic of assigning the addresses is not directly obvious. Summarizing as the bit-spacing increases the reliability and tolerance towards radiation effects it has been selected for defining the I2C addresses of DelFFi.

A Matlab script is used to create a list of addresses based on the possible 112 addresses. The difference in minimum bit spacing is shown in 3-9. Important to note is that the amount of the addresses depends on the first assigned address, with 8 (0001000) being used for DelFFi.

For DelFFi the main bus requires 10 addresses to communicate with all subsystems. As a consequence it is possible to use a bit-spacing of at least 3 bits which is shown for DelFFi in table 3-2.

Table 3-2: Assigned I2C addresses for DelFFi

Subsystem	Address [dec]	Address(bin)	Address(hex)
ADCS-MB	008	0001000	08
OBC	015	0001111	F
EPS	017	0010001	11
TRXUV1	022	0010110	16
TRXUV2	034	0100010	22
uPS	037	0100101	25
GAMALINK	059	0111011	3B
AntS	060	0111100	3C
AntS	067	1000011	43
DRB	068	1000100	44

3-3 Software Design and Architecture

The OBSW is an important of the development of the CDHS. It can be divided into a service layer, commonly known as drivers, and an application layer. Interaction between the two is shown in figure 3-10.

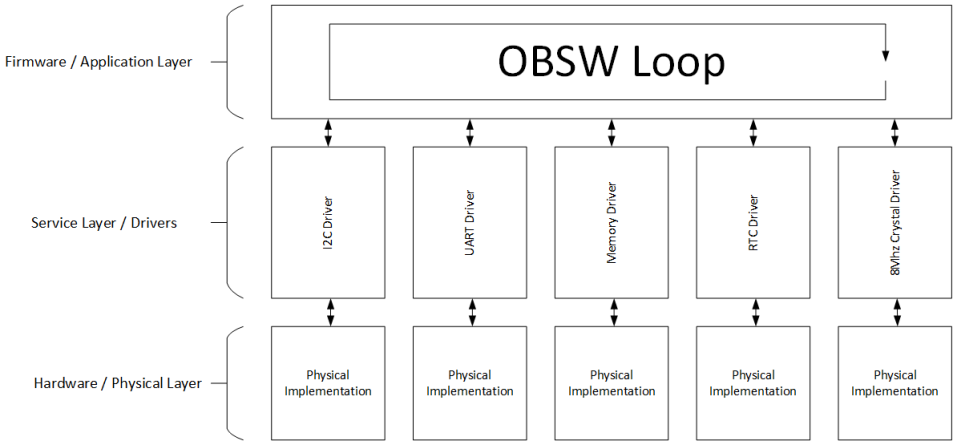


Figure 3-10: CDHS Layers

The focus of the research within this document is on the application layer architecture of the OBSW, which is defined on the basis of Delfi-n3xt and prior work. The main flow of the software is shown in figure 3-11.

3-4 Development Status

The development process of a space system can be divided into different phases. Following the classification of [18] a project starts in phase 0 in which the mission is defined. This is followed by phase A and B in which the general system architecture of the spacecraft is developed. For DelFFi summaries of the development activities can be found in [11] and [8]. The major design and implementation work is performed within phase C which is ended by

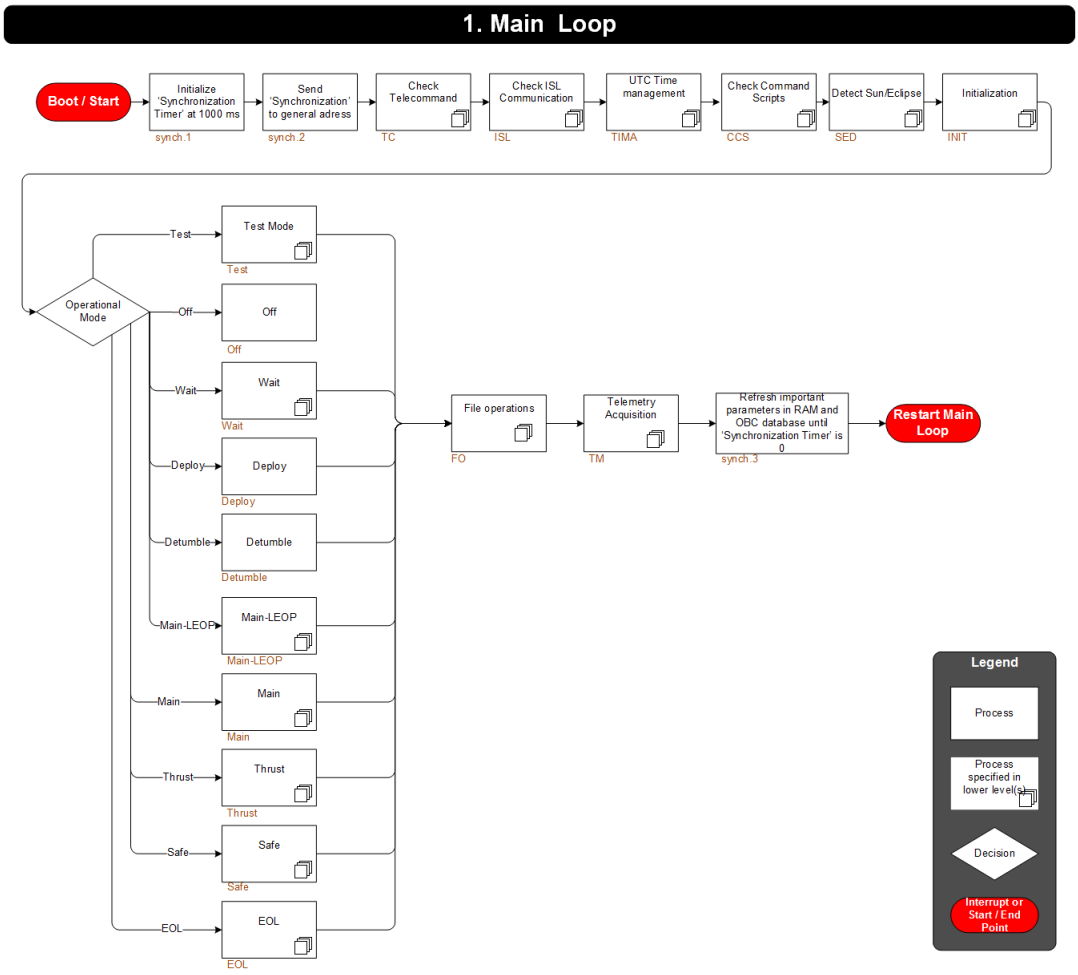


Figure 3-11: OBSW Activity Flows

the critical design review. The DelFFi project as a whole is considered to be phase D which is focused on the verification and implementation of the design. However with regard to section 3-5 the CDHS can be considered to be in phase C. An overview is shown in figure 3-12.

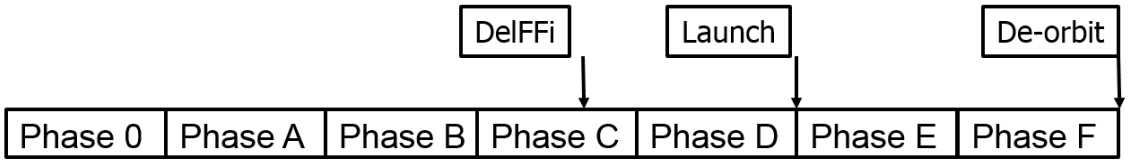


Figure 3-12: DelFFi Project Status

3-5 Remaining Work

The discussion in section 1-2 and 1-3 and the discussion within this chapter showed the status and challenges in the development of the CDHS. A technical risk analysis performed (See appendix D), showed that the Technical Readiness Level (TRL) of the hardware components ranges between 4 and 9. Opposite the TRL of the OBSW is mainly 3 and lower. Identified work packages for the further development are:

1. Increasing TRL of hardware components to 7+
2. Increasing TRL of software components to 7+
   (a) Definition and Implementation of the Application Layer for OBC and DRB
   (b) Definition and Implementation of the service layer for OBC and DRB
   (c) Integration FDIR procedures into the current architecture
3. Update and maintenance of relevant ICD
4. Engineering Model Testing
5. Improvement of the subsystem with test results
6. Flight Model Testing and Qualification

It has to be noted that the research performed within this document increases the readiness level of item 2.a and 2.c, but more work is required especially for the implementation of the software. Many of the items are currently on hold and more man power is needed to bring the development of the CDHS to a successful end.

Chapter 4

Operational Modes and States for CubeSats

For developing the system architecture of the DelFFi Command and Data Handling System (CDHS) an important task is to define the operational concept of the mission. For the CDHS development this implies to define what the capabilities (modes), physical configurations (states) of the system are at different point of times (phases). Thus the question to be answered in this chapter can be summarized as follows: Can Systems Engineering literature and industry standards be used to develop a methodology for determining the different phases, modes and states of a CubeSat mission and how can it be implemented for Delft Formation Flying Experiment (DelFFi).

The first section, 4-1, discusses the research approach for developing this methodology. In section 4-2 literature from industry and academia is reviewed and different definitions of modes, states and phases are examined. Followed by section 4-3 in which the methodology is developed and applied to DelFFi. The last section, 4-4, discusses the limitations of the developed methodology and provides conclusions and recommendations for further research.

4-1 Research Approach

Within this section the general system engineering research approach for the definition of the phases, states and modes is explained. According to [19] there is no need for a unique definition of the term Systems Engineering. Different definitions allow the use of different implementations from individuals, organizations and application areas.

While a general definition for the field of Systems Engineering (SE) might not be possible or useful, the terms are used within the development DelFFi and for Spacecraft (S/C)/CubeSats in general. As a consequence there is a need to define these terms such that every project member understands how these terms are defined and used within the project. The general approach for the research is laid out in the following paragraphs.

Question The research questions for this chapter are summarized as follows: Can different Systems Engineering tools be used to develop a methodology for determining the different phases, modes and states of a CubeSat mission and how can it be implemented for DelFFi? Thus the research should result in a methodology which is applicable for any CubeSat mission, including DelFFi. The need to answer the question arose from discussion within the DelFFi project how to define and determine the modes and states of the mission.

Observe For developing a methodology it is important to first answer how are phases, modes and states defined within the industry and academia? By reviewing industry standard and textbooks, approaches and definitions can be analysed and adapted for the formulation of the hypothesis.

Form Hypothesis From the observation a general approach for the definition of system phases, modes and states of CubeSats can be adapted and formulated. In the scope of this work only a hypothesis can be made as a suggestion to the DelFFi project and for future reference in other projects.

Perform Experiment The experiment, applying the methodology on DelFFi, is combined in the formulation of the hypothesis. This approach has been chosen to reduce the amount of iterations needed and to discuss the hypothesis on a real project instead of using arbitrary examples.

Documentation The detailed analysis is documented in this document. Additionally a abstract will be submitted to a conference at a later stage, which will focus on how to use the methodology as part of the development process of an university CubeSat.

4-2 Literature Review

This sections presents different definitions of system modes and states. The analysed literature is a limited selection of relevant information from 2 source rubrics: industry standards and general systems engineering literature. The selection rationale is first to try to identify if there are standards within the industry which define system modes and states. Followed by research into educational system engineering literature and how it defines the distinction between the terms. The selection of material is based on ease of access, not restricted by a payment, and ease of finding using an web-based search engine.

The question to be answered in this section is: How are phases, modes and states defined in industry and academia?

4-2-1 Standards

As first step it is investigated how modes and states are defined in engineering standards. A standard defining states and modes, from the field of software engineering, is MIL-STD-498.

The reasons this standard is analysed within this discussion is due to the major software component of the subsystem. The standard defines the terms mode and states as follows:

> If the system is required to operate in more than one state or mode having requirements distinct from other states or modes, this paragraph shall identify and define each state and mode. Examples of states and modes include: idle, ready, active, post-use analysis, training, degraded, emergency, backup, wartime, peacetime. The distinction between states and modes is arbitrary. A system may be described in terms of states only, modes only, states within modes, modes within states, or any other scheme that is useful. [20]

In MIL-STD-498 modes and states are considered to be arbitrary distinction and the standard is not further used. In the space industry standards are defined by for example the European Cooperation for Space Standardization (ECSS) and specific for the CDHS by the Consultive Committee for Space Data Systems (CCSDS). Within documents from both organizations system modes and states are used to describe and define aspects of systems. However no direct definition for either modes or states has been found within the released documents from ECSS and CCSDS.

4-2-2 Systems Engineering Literature

As the definition in MIL-STD-498 is ambivalent and no definition is provided by dedicated space standards, the next step is to investigate definitions within dedicated system engineering literature. A definition is provided by [21]:

> *A system mode is defined to be a distinct operating capability of the system during which some or all of the system function may be performed to a full or limited degree. Modes are the functions of the system.* [21]

Thus the author links system modes to the capabilities and functions of a system. Furthermore the author defines a system state to be:

> *State of a system is defined to be a static snapshot of the set of metric of variables needed to describe fully the systems capabilities to perform the systems function. The state of a system is the long list of variables, aka state variables.*[21]

The difference between the two definitions is that modes are based on the functions and capabilities of a system and states are a physical representation of the system expressed by a set of state variables.

The systems engineering textbook [22] has a dedicated discussion on how to define system modes and states in general. It defines a system mode to be:

> *"An abstract label applied to a collection of system operational capabilities and activities focused on satisfying a specific phase objective"* [22]

Similar to the definition of system modes by [21], both authors define system/operational modes to be related to the capabilities of a system. Furthermore states are defined by the author as:

"The operational or operating condition of a SYSTEM OF INTEREST (SOI) required to safely conduct or continue its mission. For example, the operational state of an aircraft during take-off includes architectural configuration settings such as wing flap positions, landing gear down, and landing light activation." [22]

The author of [22] proposes a process for determining states and modes, starting by defining the system phases. A mission can be divided into 3 abstract phases of operation: The pre-mission-phase, the mission phase and the post-mission phase. These phases can be further differentiated into sub-phases if required.

After defining the phases the author proposes to assign the use-cases of a system to the three general phases. A use case of a system is defined by the author to be how the user will operate the system to achieve a defined outcome. In case the use cases have shared objective or outcome it is possible to cluster these into system modes.

With the defined sets of modes it is important to determine how the system transits from one mode to the other. Pre-defined events and conditions are used to initiate a transition from one mode to the other. Such a trigger event can be initiated externally, for example by ground, or predefined conditions, such as drop to a safe mode.

Both examples from system engineering showed that modes are related to the capabilities and use cases of a system. They express what a system should achieve and how it can be utilized. Both authors define states to be a physical representation of a system. Within [22] a general approach is described of how to define phases, modes and states.

4-2-3 Conclusions

The question to be answered in this section is how system modes, states and phases are defined within academic literature and standard. A review of space standards from ECSS and CCSDS shows that are no definitions available from these organizations. A standard used within software engineering, MIL-STD-498, provides only an arbitrary distinction between the terms.

Within the investigated SE literature different ,albeit similar, definitions are found for phases, states and modes. System modes are commonly defined to be a expression of the required capabilities and use cases of a system. The transition from one mode to the other is defined by entry and exit criteria.

A state is defined to be the physical representation of a system at certain point of time. This means a state express all physical system characteristics such as active subsystems, received power and used bandwidth.

Finally phases are defined to be an expression of the life cycle of the system. Following the discussion in [22] 3 phases for every project can be identified. These are the pre-mission, mission and the post-mission phase. It is possible to divided these phases further into sub-phases.

Answering the question shows that general systems engineering literature provides definitions and approach, which can be used to develop a specialized methodology for CubeSats. In the following discussion the methodology as proposed by [22] is adapted to CubeSats and implemented for DelFFi.

4-3 Methodology for CubeSat System Phases, Modes and States

Section 4-2 concluded with a description of how modes, states and phases are defined within literature. From the general methodology, as described in [22], a series of steps is developed of how to define system modes, states and phases for a CubeSat program. A general flow down of this process is shown in figure 4-1.

First the different stakeholders of a project need to be analysed. From the identified operational stakeholders and the mission objectives use cases and phases of the mission can be defined. With the use cases assigned to the system phases, it is possible to identify the system modes. The final steps are the definition of the system mode transitions and the definition of the system states. For further improvement the results of the process need to be iterated and discussed with other team member to create a definition of system modes, states and phases which is accepted by all team members. The results of this research have been presented to the DelFFi team on the 17th of April and are planned to be implemented in the project.

The question to be answered in this section is how can the common core of the definitions found in section 4-2 and the approach as discussed in [22] be used to define system phases, modes and states of a CubeSat and be implemented for DelFFi?

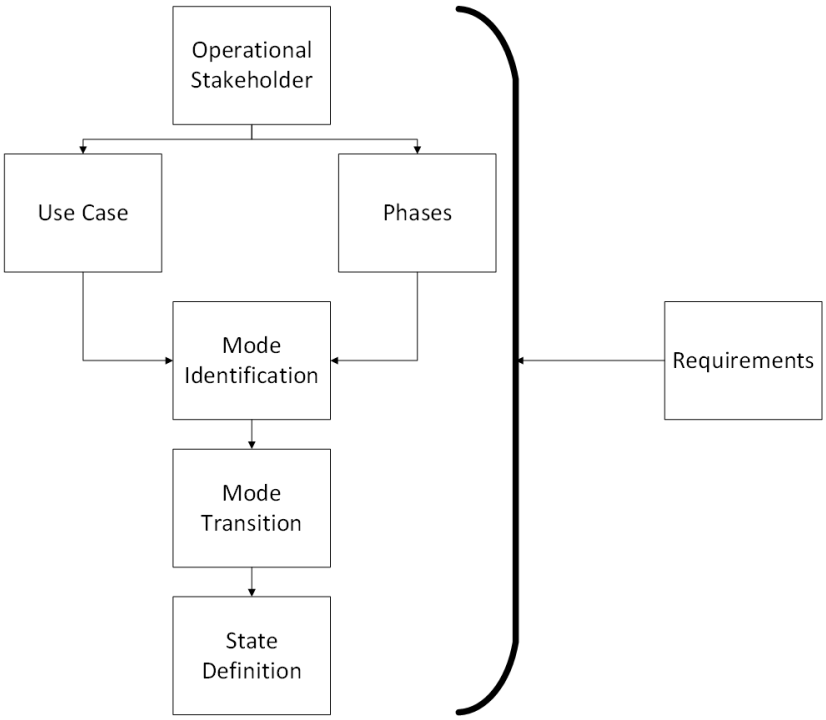


Figure 4-1: Development Flow down for the definition of System Phases, Modes and States

4-3-1 Stakeholder Analysis

For starting the process it is proposed to determine the stakeholders of the system. The aim of the analysis is to identify actors, which have an operational influence on the system. By

identifying the actors the use cases can be determined in section 4-3-3. A tool to determine the stakeholder is an exploration tree. Such a tree is shown for the DelFFi mission in figure4-2.

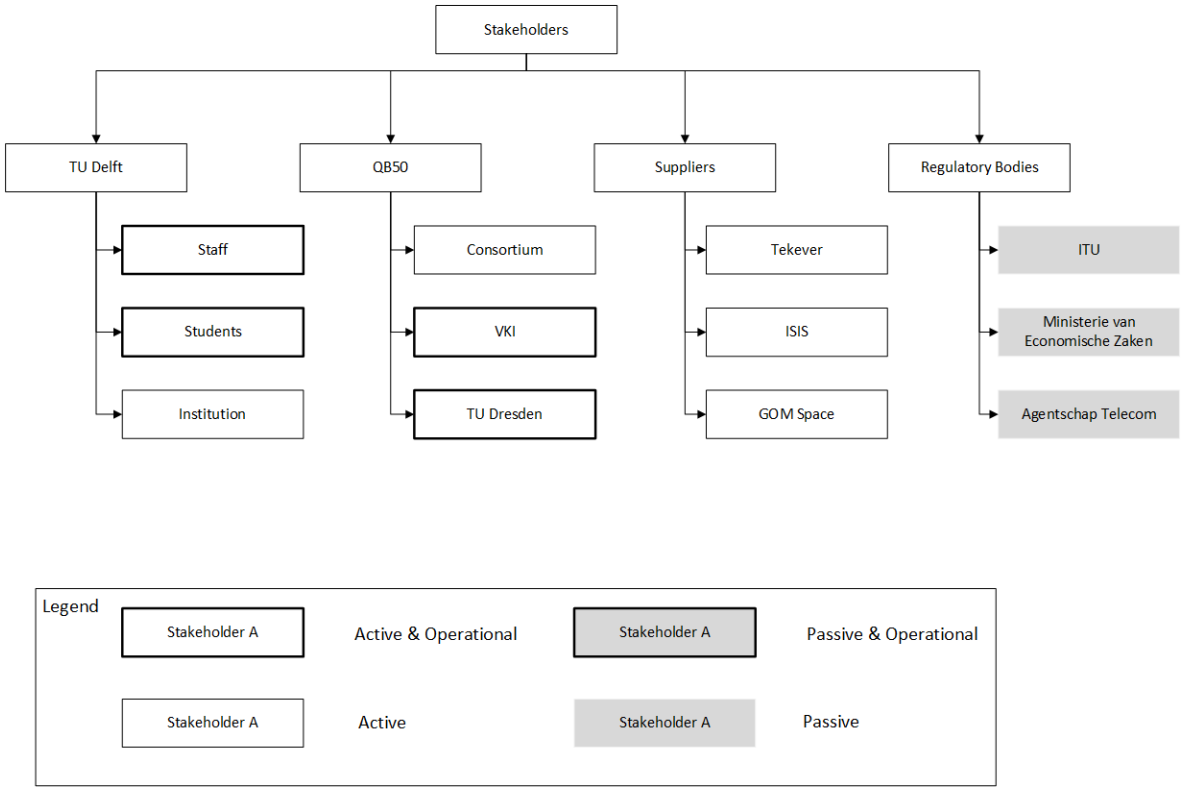


Figure 4-2: Stakeholder Exploration tree

The figure differentiates between active and passive stakeholders. An active stakeholder interacts with system directly and passive are influencing the system not by standards or regulations. Furthermore, stakeholders further differentiated into if a stakeholder has impact on the operations of the spacecraft. This limits the analysis to stakeholders who interact or influence the system during the operations of the system. For example component suppliers are active, but in most cases, except for technology demonstration, do not influence the system during the operations.

4-3-2 System Phase definition

The second step is to define the system phases of the mission. Information used are taken from general discussion about the spacecraft life-cycle [18], mission objectives (see [7] and [11]) and regulations. The relation between project phases and system phases are shown in in figure 4-3.

The pre-mission phase can be divided to on-ground and Launch and Early Operation Phase (LEOP). The start of the system phases is the first assembly of the majority of spacecraft components, in general the first full engineering flat-sat test, which occurs in phase D of the project and ends after the conclusion of LEOP.

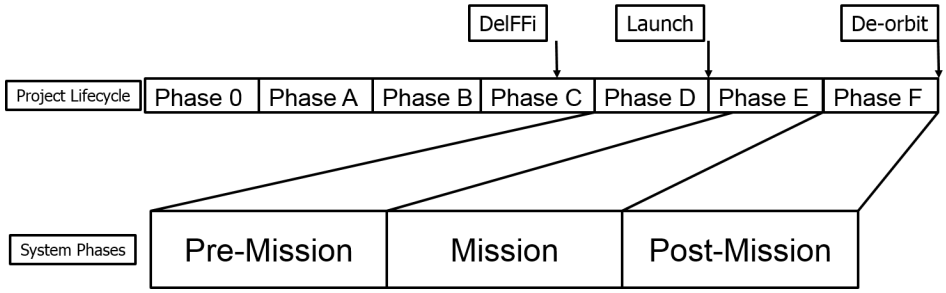


Figure 4-3: Project Phases and System Phases

Following is the mission phase, which begins when the system is commissioned and can be divided into several sub phases. For example a mission could have a cruise phase and observation phase. A more interesting case if the satellite has several objectives in Earth orbit, such as DelFFi which has to demonstrate formation flying and observation of the upper atmosphere (see chapter 2). The challenge for DelFFi is if the objectives are mutually exclusive or can be achieved simultaneously. There are two general approaches: one is to define the phases depending on the active science payloads, hence using a more bottom-up approach or defining the phases based on the primary objective.

For determining which of the two approaches to use, a first step is to estimate the durations of the different phases. A major consideration to take into account for DelFFi is the time needed for both S/C to reach the required 1000 [km] separation distance.

DFF-MIS-X-XX: The formation flying shall be demonstrated after the nominal along-track distance is no less than 1000 km. [3]

The time for reaching the distance can be estimated using equation 4-1.

$$t_{sep} = \frac{s_{sep}}{v_{diff}} \tag{4-1}$$

The term s_{sep} is the required separation distance between the two S/C and v_{diff} is the absolute velocity differential in along track direction. In an early mission analysis [11] the value of 1 $[\frac{m}{s}]$ is considered. However [23] shows that the deployer used for Atmospheric Research Network (QB50) can provide a ΔV of upto $2\frac{m}{s}$. Assuming a ejection in opposite directions the maximum possible ΔV is $4\frac{m}{s}$. This gives a possible range of velocities of 3 $\frac{m}{s}$, however as the deployment can be tuned a velocity differential of $1\frac{m}{s}$ is assumed as nominal value.

Using both values the expected time before reaching separation distance is within a range of minimal 2.8 and nominal 11.6 days. This estimation assumes linear motion, although the differences in along-track velocity will result in small differences in the eccentricity of the orbits. For longer time spans this estimation might not be valid and numerical simulations should be used.

With reaching the along-track separation of 1000 [km] a drift-stop manoeuvre has to be performed. [7] According to [24] the propulsion system on both spacecraft should provide a mission ΔV of $15\frac{m}{s}$. Assuming the expected velocity difference and a margin, $13.9\frac{m}{s}$ will

remain for formation keeping. With the required ΔV of $0.15\frac{\frac{m}{s}}{day}$ [7] a formation flying time of 92 days is possible, with a minimum of 70 days for the maximum velocity differential.

The following step is to determine the remaining time before re-entry into the atmosphere. A first order approximation can be achieved by determining the ballistic coefficient, the expected solar activity and the use of standard tables.

$$\beta = \frac{M}{C_d A} \tag{4-2}$$

$$A = \frac{1}{2}\sum_{i=1}^{n} S \tag{4-3}$$

In equation 4-2 βB is the ballistic coefficient, C_d the drag coefficient and A the surface area in flow direction assuming a free tumbling S/C. The variables denoted with S are the areas of the different surfaces of the S/C with the dimensions given in [25]. Both [7] and [26] estimate the drag coefficient of a S/C to range between 2.2 and 2.3.

$$\begin{aligned} A &= \frac{1}{2}[4S_{bodywalls} + 4S_{wings} + 2S_{top/bottom}] \\ &= 2 * 0.0825[m] * 0.317[m] + 2 * 0.0825[m] * 0.317[m] + 0.1[m] * 0.1[m] \\ &= 0.11[m^2] \end{aligned} \tag{4-4}$$

The mass, M, of one of the S/C is 3.639 [kg][27], resulting in a ballistic coefficient of 14.4 $[\frac{kg}{m^2}]$ for a free tumbling spacecraft. For achieving the mission objective the DelFFi satellites are required to point in velocity direction, which reduces the area in the flow to 0.01 $[m^2]$, resulting in a ballistic coefficient of 158.2 $\frac{kg}{m^2}$ Furthermore the predicted solar activity for a mission in 2016 shows that the solar activity is near its cycle maximum. [28] On the other hand the current cycle is weaker, as shown in 4-4.

Finally the table shown in 4-5 can be used to determine the orbital decay. The reference [29] has been published in 1999 with a normal solar cycle. Contrary the current solar cycle has a smaller amplitude and it is assumed that the orbital life time is half the value of min and max. In case the for a 300x300 orbit this coincides with a mission time as discussed in [11] and formation flying can be tested for most of the mission duration. However for orbits which are higher, the propellant on-board will be depleted and the formation flying will not be possible for the full mission duration.

Mission Analysis shows that the time for both S/C to be separated by 1000 km is within the range of 3 to 12 days depending on velocity differential after ejection from the launcher. For determining the required manoeuvre the position and velocity of the S/C needs to be known. According to [11] this can be achieved with the use of Two-Line Element (TLE) with an uncertainty of less than 1 [km] [30]. As a consequence the uncertainty in velocity should be negligible. A more severe issue is the matching of TLE to corresponding CubeSats. Shortly after deployment it might not be possible to discern which TLE element belongs to which CubeSat. It is proposed to further investigate this in scope of the formation flying development and add a requirement.

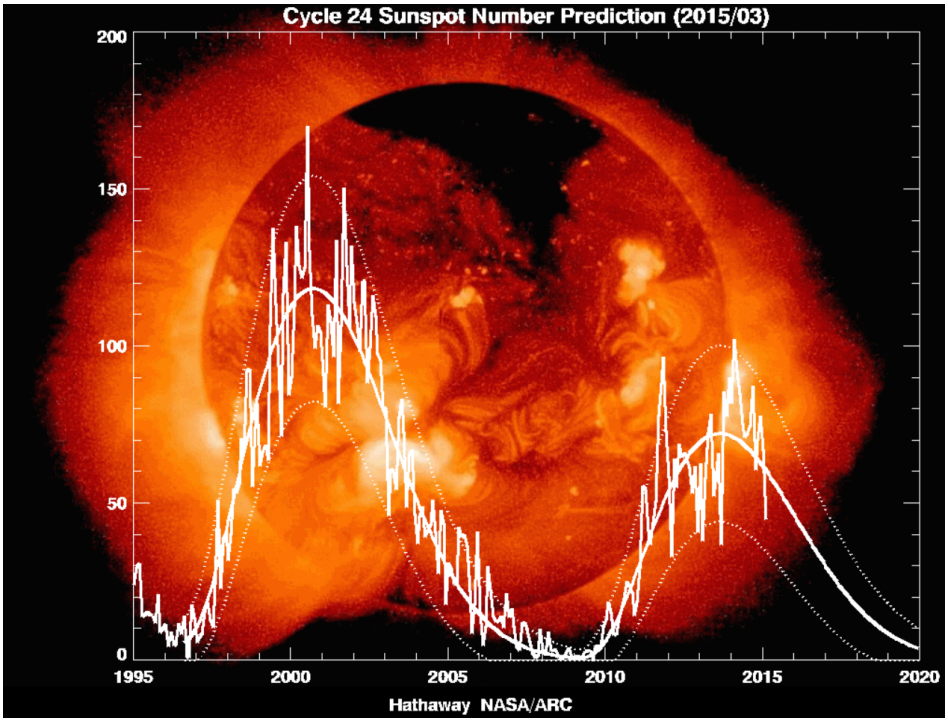


Figure 4-4: Past Solar Activities and Predictions (from [28])

Alt (km)	ORBIT DECAY RATE				ESTIMATED ORBIT LIFETIME			
	Solar Min 50 kg/ m^2 (km/yr)	Solar Max 50 kg/ m^2 (km/yr)	Solar Min 200 kg/ m^2 (km/yr)	Solar Max 200 kg/ m^2 (km/yr)	Solar Min 50 kg/ m^2 (days)	Solar Max 50 kg/ m^2 (days)	Solar Min 200 kg/ m^2 (days)	Solar Max 200 kg/ m^2 (days)
0	3.82×10^{13}	3.82×10^{13}	9.55×10^{12}	9.55×10^{12}	0.00	0.00	0.00	0.00
100	1.48×10^{7}	1.64×10^{7}	3.70×10^{6}	3.96×10^{6}	0.06	0.06	0.06	0.06
150	5.30×10^{4}	6.58×10^{4}	1.32×10^{4}	1.57×10^{4}	0.24	0.18	0.54	0.48
200	5.75×10^{3}	1.14×10^{4}	1.44×10^{3}	2.67×10^{3}	1.65	1.03	5.99	3.6
250	1.09×10^{3}	3.45×10^{3}	2.72×10^{2}	7.99×10^{2}	10.06	3.82	40.21	14.98
300	2.67×10^{2}	1.29×10^{3}	6.67×10^{1}	2.95×10^{2}	49.9	11.0	196.7	49.2
350	7.64×10^{1}	5.44×10^{2}	1.91×10^{1}	1.23×10^{2}	195.6	30.9	615.9	140.3
400	2.40×10^{1}	2.48×10^{2}	6.01×10^{0}	5.50×10^{1}	552.2	77.4	1024.5	346.9
450	8.12×10^{0}	1.19×10^{2}	2.03×10^{0}	2.60×10^{1}	872	181	1,497	724
500	2.97×10^{0}	5.95×10^{1}	7.42×10^{-1}	1.28×10^{1}	1,205	393	2,377	3,310
550	1.20×10^{0}	3.07×10^{1}	3.01×10^{-1}	6.53×10^{0}	1,638	801	5,470	4,775
600	5.60×10^{-1}	1.63×10^{1}	1.40×10^{-1}	3.41×10^{0}	2,580	3,430	14,100	13,400
650	3.05×10^{-1}	8.83×10^{0}	7.64×10^{-2}	1.83×10^{0}	5,560	4,550	28,500	27,900
700	1.92×10^{-1}	4.92×10^{0}	4.81×10^{-2}	1.00×10^{0}	13,400	12,600	53,400	52,700

Figure 4-5: Satellite Orbit Decay reference table (from [29])

Consequently the LEOP phase is constrained to 12 days, followed by the start of the mission phase. The DelFFi requirements [3] include the definition of a phase:

DFF-MIS-X-XX: Between LEOP and formation flying, there shall be a formation acquisition phase leading to the desired formation geometry [3]

It is proposed to delete this requirement, as it is the only requirement relating to a phase of the mission. However if a detailed sub-phase planning is required, formation acquisition can be assigned to the pre-mission phase.

Finally formation flying is a continuous process and experiment can not be stopped and continued at a later time. Thus the first phase in the mission phase is designated the nominal operations phase. This means that both mission objectives, atmospheric measurements and formation flying, can be achieved at the same time. First order analysis shows formation flying is limited to a maximum of 92 days, which for the original assumed orbit is close to the expected life time.

CubeSats, such as DelFFi, are using launch ride-shares, which implies that there is an uncertainty about the orbit parameters. Thus the orbital lifetime of both, Delta and Phi, can be significantly longer than the maximum possible time for formation flying. As a consequence an extended operation phase is planned after the end of the formation flying. The length of this phase is dependent on the re-entry and the continued interest in operating the S/C.

Concluding the approach of defining 2 distinct phases is chosen. The first phase nominal operations will achieve both mission objectives, while the second extended operations will be limited to the operation of the science payload.

In the post-mission an End of Life phase is scheduled which covers the time from decommissioning to the de-orbiting of the S/C. A reason for deactivating the satellites before de-orbiting can be the regulations to insure the satellites while active. [31]

A summary of all system phases is shown in 4-1. The first three phases are defined using project planning.

Table 4-1: DelFFi Mission Phases

Archetype Phase	Subphase	Time span	Description
Pre-Mission			
	System Development	From 2013 to 01.05.2015	Phase A-C of the project, development of the different subsystems.
	Testing	From 01.05.2015 to [tbd]	Testing of the integrated satellites
	Delivery/Prelaunch	[tbd] to 2016/2017	Delivery to Launch provider, Integration into CubeSat deployer and transport to the launch site
	LEOP	11 days	Launch and commissioning of the S/C
Mission			
	Nominal Operations	92 days	Time for fulfilling formation flying objective
	Extended Operations	0 to[TBD]	Possible continued operations of S/C and science payload.
Post-Mission			
	End of Life	[TBD]	Time after decommission of the two satellites

4-3-3 Use-Case Definition

Following the definition of stakeholders in 4-3-1, the use cases of the different stakeholders can be identified and assigned to the defined phases.

According to [22] a use case can be defined to be the expression of how a user wants to exploit a system to achieve a desired outcome. Furthermore an actor on a system is defined by Unified Modeling Language (UML) to be an entity which interacts with the system but is not part of it. An actor can be entities from organization level, for example ESA, to personal, such as an operator but also can be other systems, unauthorized users, threats or environmental conditions. The first step in defining the use cases is to identify the actors in the system from the stakeholders. Actors can use, or operate the system, to achieve a desired outcome. The defined operational stakeholders in 4-2 are actors on the system

Additionally to the operational stakeholders actors can be other systems acting on the system, for example a ground segment on a spacecraft, environmental effects and regulatory bodies.

For DelFFi the actors, determined from the operational stakeholders, are: TUD Staff, TUD Students, QB-50 VKI and QB50 TU Dresden. Further stakeholders taken into account are regulatory bodies, which the main in case of DelFFi are the International Telecommunication Union (ITU) and the Dutch government.

The operational stakeholders want to operate the system to achieve a certain use or outcome. Corresponding use cases can be derived from the objectives and needs of the stakeholders.

TUD Students have the main objective to use the system to graduate from university by performing research and documenting it in thesis and academic publications. For achieving these use cases, the system needs to be designed, qualified, commissioned and operated.

TUD Staff uses the system to teach and educate students on a real space system. Next to the education use case the actor uses the system for research which includes the publication of papers. For the education and research the actor has the wished outcome of a designed, qualified, commissioned and operated spacecraft. The mission objectives of the system are defined by this actor in [11].

QB50 VKI As initiator of the QB50 project Van Karman Institute (VKI) wants to use the system to perform a atmospheric research and facilitate the development of CubeSats at universities. [32]

QB50 TU Dresden As system developer of the scientific payload FIPEX the actor uses the system to perform atmospheric research.

The identified operational use cases of the system can be collected in a table, which is shown for DelFFi in table 4-2.

Following the interaction between the use cases and different actors can be mapped using a use case diagram on the basis of UML. Such a diagram is shown in figure 4-6 for DelFFi. Within

Table 4-2: DelFFi Operational Use Cases

Use Case	Description
Design/Qualify Satellite	Assuring on-ground that all subsystem full fill the requirements (verification) and work as intended (validation).
Commission Spacecraft	Prepares the spacecraft for the achieving the main mission goals and validating all subsystem in orbit. Includes use cases such as deploy appendages and detumbling.
Perform S/C operations	Operating the spacecraft(s) from ground segment to execution of the on-board software.
Perform formation flying	Controls the distance between the two CubeSats Delta and Phi. Includes the use cases of perform orbital manoeuvres and control formation.
Take Atmospheric Measurements	Operate the Flux-(Phi)-Probe-Experiment (FIPEX) payload and transmit science data to the ground.
Perform Error Handling	In case of detecting conditions outside the operational envelope start mitigation measures.

the figure the relation between actors and use cases has only been indicated for TUD-staff and students for better readability.

Use cases can include and extend other use cases, with difference that the former is a sub use case which is always part of the upper level use case and the latter a possible sub use case which is not strictly necessary. For example a sub use case of "perform spacecraft operations" could be "execute on-board software", which has a possible extension with "perform error handling".

The next step is to assign the use cases to the system phases. In general all activities/uses of the system which are required for preparation to achieve the mission are assigned to the pre-mission phase, all use cases related to the achieving the mission of the system are allocated to the mission phase and finally all use case after completing the mission objectives are allocated to the post mission phase. The allocation can be documented in a table as shown in table 4-3.

4-3-4 System Mode Identification

With the defined use cases in 4-3-3 the system modes can be determined. A mode is the expression of a capability of the system. It can be expressed by a single use case or a combination of different use cases which share are a capability or outcome. Modes are limited to a archetypical phase, hence use cases in pre mission and mission can not be combined into a single mode but are separate for each archetypical phase. A tool which can be used is a phase - use case diagram in which the different use cases can be visually grouped into modes. For DelFFi this diagram is shown in figure 4-7

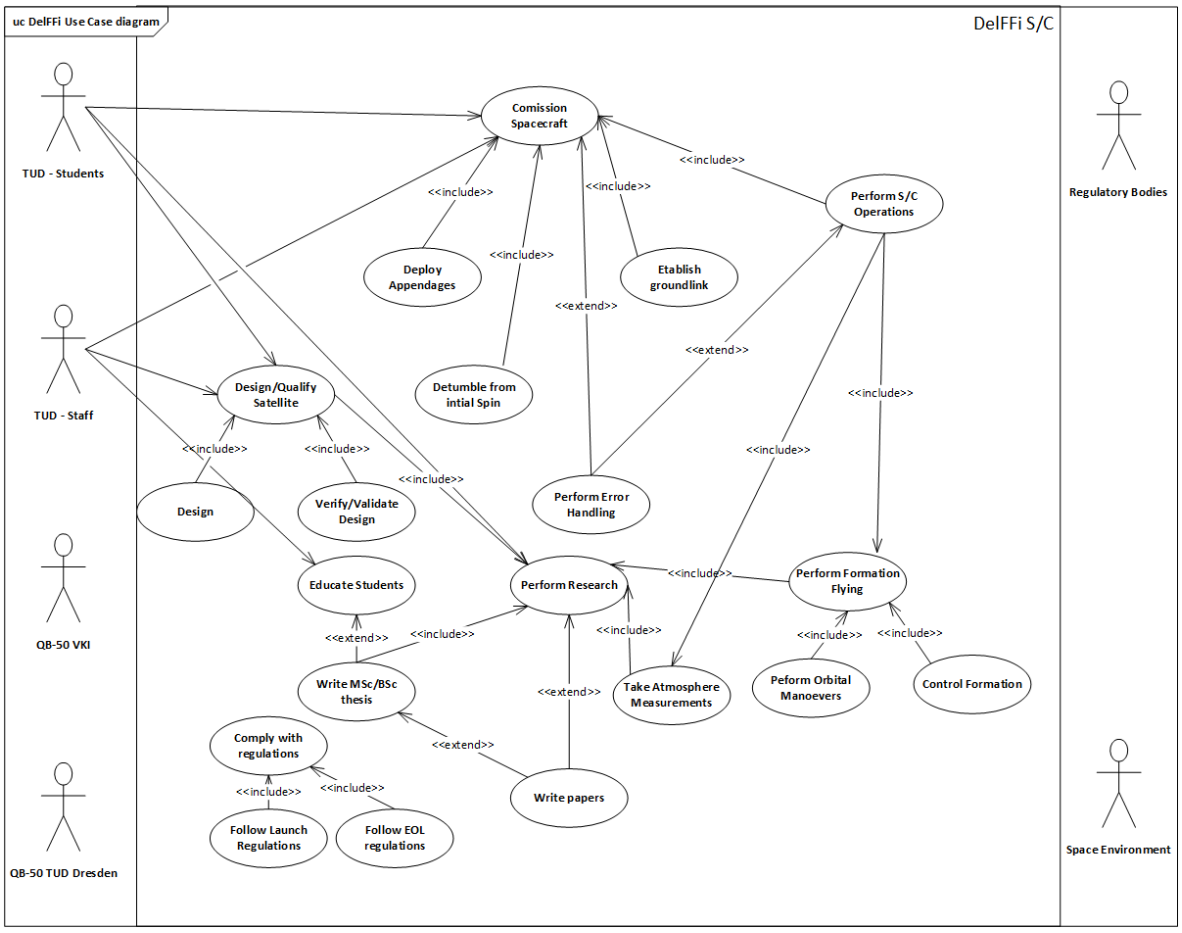


Figure 4-6: DelFFi Use Case Relation Diagram

Table 4-3: Use Case Phase Assignment

Pre-Mission		Use Cases
	System Development	Educate Students, Perform Research, Design/Qualify Satellite
	Testing	Design/Qualify Satellite
	Delivery	
	LEOPS	Commission Spacecraft
Mission		
	Science Observation	Take atmospheric measurements, Perform S/C operations
	Formation Flying	Perform Formation Flying, Perform S/C operations
Post-Mission		
	End of Life	

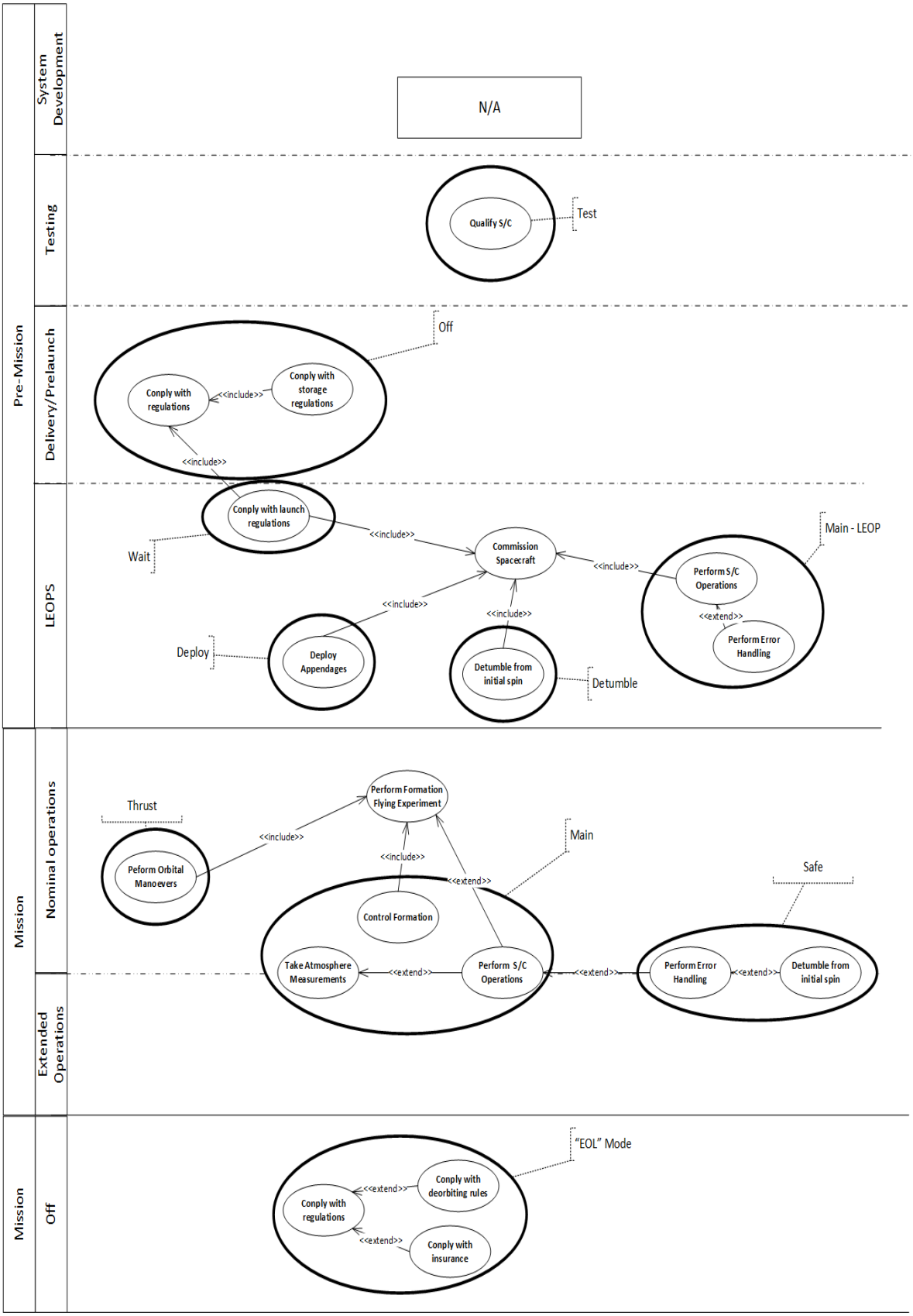


Figure 4-7: General Use case diagram for Space Missions

The modes can be collected into a table, as shown for DelFFi in 4-4. In the table the main characteristic of each mode is described.

Table 4-4: DelFFi System Modes

Phase		Mode	Description
Pre Mission	Testing	Test	Mode use on ground for qualification of the satellite.
	Delivery/Pre-launch	Off	Mode used from storage till separation distance from the launch vehicle
	LEOP	Off	Identical to Off in Delivery/Pre-launch
		Wait	Waiting for sufficient separation distance
		Deploy	Deployment of the S/C appendages
		Detumble	Reducing S/C roll rates in xyz and Velocity pointing
		Main-LEOP	Commissioning of payloads and drift wait
Mission	Nominal Operations	Main	Routing operations of the S/C and payloads
		Thrust	Used for the capability to perform and prepare for orbital manoeuvres.
		Safe	In case of failures the spacecraft is secured to allow recovery using FDIR or on ground
	Extended Operations	Main	Same as in Nominal Operations
		Thrust	Same as in Nominal Operations
		Safe	Same as in Nominal Operations
Post	End of Life	EOL	No emission of radio signals

4-3-5 System Mode Analysis and Transition Definition

With the determined modes in 4-3-3 two more steps are required to complete the definition of the system modes. First the identified modes need to be analysed and iterated with the system states (see section 4-3-6) and secondly the transition between the different modes needs to be defined.

Mode Transitions

The different modes in a phase will form a network from which it is possible to transition from one to the other. For the LEOP sub phase this is shown in figure 4-8 and for the nominal and extended mission in 4-9.

For documentation all modes and their corresponding entry and exit criteria for DelFFi are summarized in the following discussion.

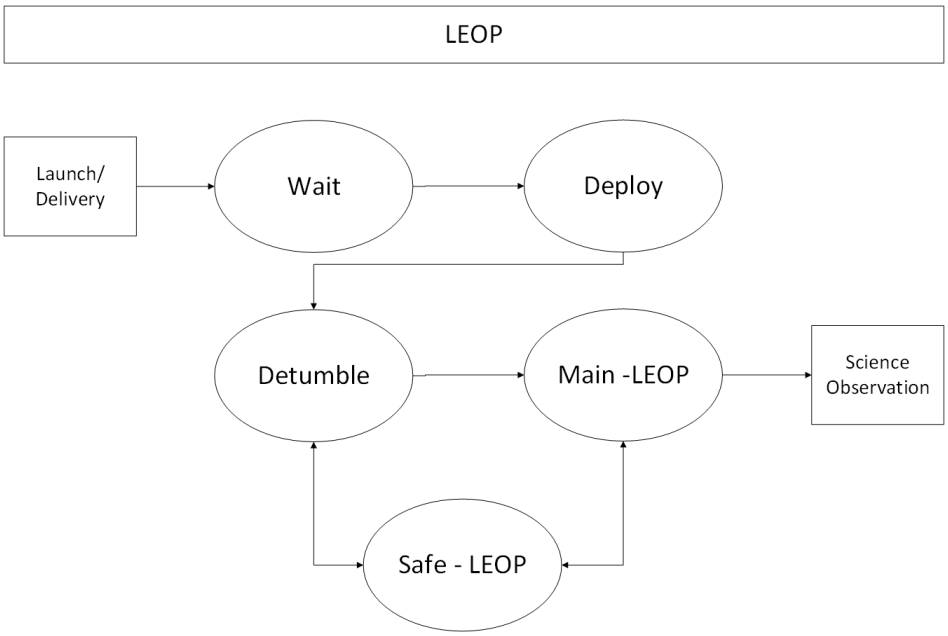


Figure 4-8: System Mode with activation and exit criteria.

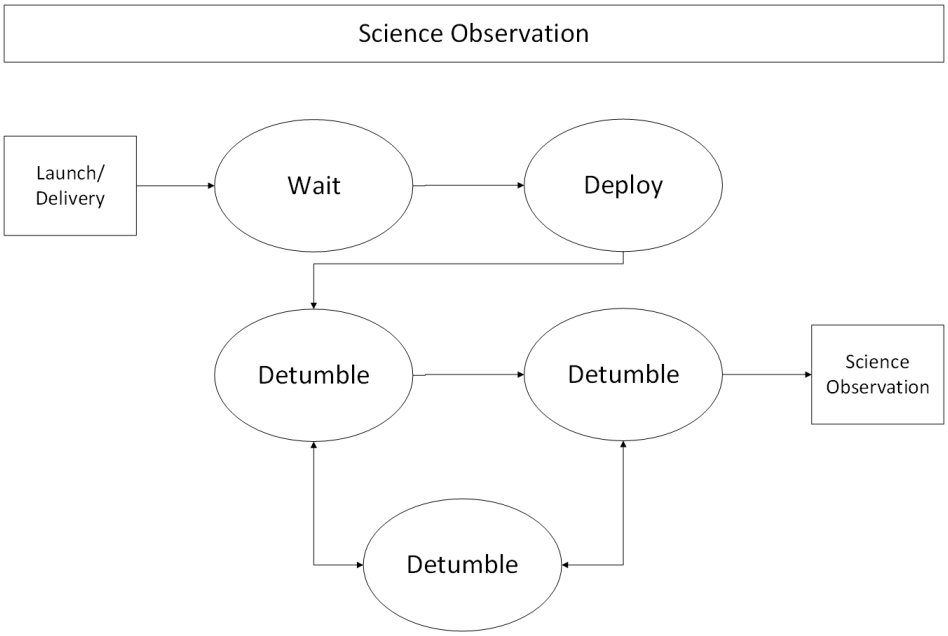


Figure 4-9: System Mode with activation and exit criteria.

System Mode - Test

Phase(s)	Testing
Main Capability	Allow verification and validation of system design
Duration	[TBD] days

The main use case of the test mode is to allow the qualification of the system on ground. In general there are several stages in testing from single subsystem test during development in phase A to C to integrated satellite tests in phase D. [18] From a system perspective test mode starts when the S/C transitions from the system development phase to the testing phase. The activation and exit criteria are shown below.

Transitions to	**Activation Criteria**	**Exit Criteria**
Off Mode	User defined	User defined

In section 4-3-6 no states for this mode are defined, as the physical configuration of the system might vary during testing.

System Mode - Off

Phase(s)	Delivery/Prelaunch and LEOP
Main Capability	Do not interfere with storage, deployer and launcher
Duration	[TBD] days

The off modes full fills the requirement for storage of the S/C on ground and during launch.

Transition to	**Activation Criteria**	**Exit Criteria**
Wait	User defined	Power switched on

During storage and launch the S/C needs to be inactive, hence powered down. A single state is required for this mode.

System Mode - Wait

Phase(s)	LEOP
Main Capability	Do not interfere with launcher and QB50 deployment sequence
Duration	20+ [s]

The wait modes is required by regulations from the launch provider. After ejection from the launch dispenser, both S/C have to remain in wait mode for at least 20 seconds, assuming a launch by the Europeanized Soyuz [33]. Due to the deployment of upto 50 CubeSats by QB50 and possible primary payloads a higher wait time might be necessary to ensure sufficient

distance. This is dependent on the deployment strategy and is outside of the scope of this discussion.

Transition to	**Activation Criteria**	**Exit Criteria**
Deploy	Power switch = on	Software Counter reaches T = 20s + x

After ejection from the deployer, the power switch of the Electronic Power System (EPS) is turned on and the S/C is powered. The On-board Computer (OBC) will boot and countdown the wait time before sending commands to any other subsystems.

System Mode - Deploy

Phase(s)	LEOP
Main Capability	Deploying of all appendages of the S/C
Duration	[TBD] [min]

After the separation from the launch vehicle and waiting for sufficient time the first objective is to deployment of the appendages to allow communication with the ground and deployment of the solar cells on the wings. The duration needs to be determined by analysing the time required to wait after each deployment. An important factor is the order of deployment. There are 2 factors to be considered: the need to establish ground link and the movement of the appendages during deployment.

The power budget (see appendix C-1) shows that 3.9 [W] are available before deployment of the solar panels, assuming one panel in the sun. [34] For ground link the communications system needs to active in the transceiver sub mode which results in a power consumption of 2.25 [W]. If only uplink, to command to retry deployment, is required the power demand is 0.6 [W]. This shows that it is possible to have an established ground-link without the deployment of the solar array wings.

For the movement of the appendages the dynamic range of antennas is shown in figure 4-10. The figure shows that both appendages are within their respective dynamic ranges. As the antennas are more flexible, it is advised to first deploy the solar panels and confirm and provide sufficient time to get into their final position and in a second step deploy the antennas. Due to their length and thinness the antennas are more flexible. Additionally their shape provides stiffness in the z-direction of the spacecraft and oscillation in z should be significantly lower than in x/y. This needs to be confirmed by analysis or testing.

According to [35] the deployment of each antenna has duration of 4 [s]. The time needed for deployment of the wings is to be determined in further analysis.

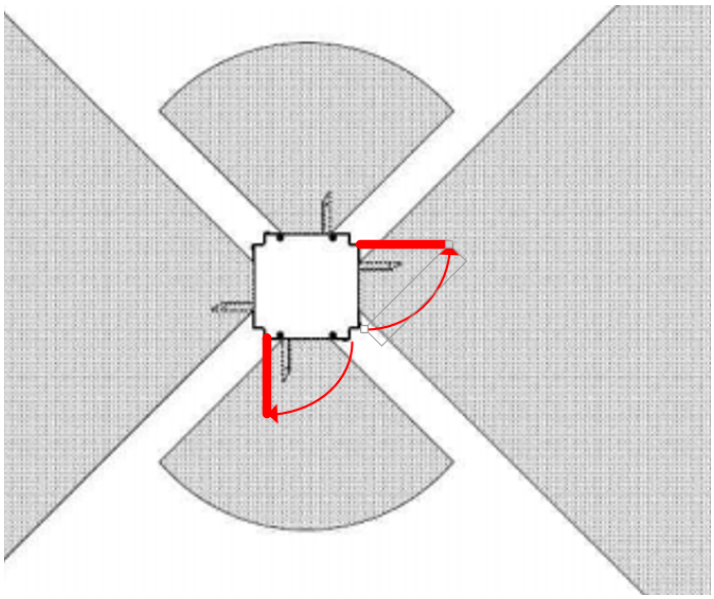


Figure 4-10: Dynamic Movement Range Antennas with indicated Solar Panel Deployment (adapted from [35])

Transition to	Activation Criteria	Exit Criteria
Detumble	Software Counter reaches T = 20s + x	Deployment confirmed User commanded

The deployment mode will have two distinct states: the first is the deployment of the solar panels and the second is the deployment of the antennas. An argument can be made to include a wait mode, in case not all deployments can be confirmed.

System Mode - Detumble

Phase(s)	LEOP
Main Capability	Reduction of initial tumble rates
Duration	0.5 days

After deployment from the launcher, both S/C will have an initial spin rate. According to [36] the S/C shall be able to recover from an initial spin rate of $\pm$ 50 [deg] and recommends to plan for non-nominal spin of up to $\pm$ 90 $[\frac{\text{deg}}{s}]$. The current Attitude Determination and Control System (ADCS) design assumes to be able to detumble from $\pm$ 45 $[\frac{\text{deg}}{s}]$, which is slightly below the requirement from QB50. (See [37] and Appendix A-1). It is assumed that detumbling of the satellites is completed within 6.75 orbits corresponding to 0.42 days.

Transition to	Activation Criteria	Exit Criteria
Main-Leop	Confirmed deployment User commanded	Angular velocities below 1 $\frac{\text{deg}}{s}$ User commanded

The detumble mode has several supporting states, in general there will be distinct states for eclipse and sun and states depending on the design of the ADCS.

System Mode - Main-LEOP

Phase(s)	LEOP
Main Capability	Preparing the satellites by commissioning and in-situ testing of the subsystems
Duration	11.6 days

After completion of detumbling, the final objective within the LEOP phase is to commissioning the S/C in orbit before the formation stop manoeuvre has to be performed. These activities can include the dry tests of the propulsion system, testing of the science payload and the testing of up- and downlink to earth.

The duration of this mode is limited by time required to detumble and the required time before the formation acquisition manoeuvre can be performed. Assuming $1\frac{m}{s}$ velocity differential the duration of the mode is 11.5 days.

Transition to	Activation Criteria	Exit Criteria
Main	Confirmed Detumbling User commanded	User Commanded

As the mode is used to test all aspects of the spacecraft, different states are required to allow in-situ testing of each subsystem. Also the mode ends with the formation acquisition manoeuvre which needs to be supported by the corresponding propulsion states.

System Mode - Main

Phase(s)	Nominal and Extended Operations
Main Capability	Operation of S/C and payload
Duration	92 days + X

In the main mode during nominal and extended mission the uses cases are the standard S/C operations and the utilization of the payloads. According to [36] there are observation periods in which FIPEX payload needs to be active. For completing the formation flying objective in the nominal operation phase, the ADCS is executing the formation flying algorithms. [9]. Furthermore it is possible to operate the GAMALINK payload as inter-satellite link between both S/C. The mode can within the mission phase transition to the thrust mode and the

safe mode. As a consequence its duration is dependent on the number thrust manoeuvres in nominal phase, possible safe mode drops and the time to de-orbit or decommissioning of the satellite.

Transition to	**Activation Criteria**	**Exit Criteria**
Thrust	Completed Thrust	User Commanded
Safe	User commanded(from Main-LEOP)	Thrust Commanded
EOL	User commanded(from Safe)	Safe Mode Drop

The main mode is supported by several states to allow the successful operation of the different payloads. This includes differences in sun and eclipse, operation of FIPEX and GAMALINK in different configurations and the execution of formation flying algorithms

System Mode - Thrust

Phase(s)	Nominal and Extended Operations
Main Capability	Perform orbital manoeuvres
Duration	n*($t_{preheat}$+$t_{thrusting}$ [min]

For achieving the objective of formation flying the S/C needs to able to perform thrust manoeuvres. The required thrust time is determined by the Formation Flying (FF) algorithms in main mode. Upon calculating a orbital manoeuvre the algorithms will define a set time and further variables. If the time is reached the main mode will autonomously transition to the thrust mode. The duration of a thrust mode instance is defined the time required to pre-heat the propulsion system and the maximum thrust time allowed by the ADCS. Currently these times are analysed from a propulsion system and formation flying concept perspective.

Transition to	**Activation Criteria**	**Exit Criteria**
Thrust	Completed Thrust	User Commanded
Safe	User commanded(from Main-LEOP)	Thrust Commanded
EOL	User commanded(from Safe)	Safe Mode Drop

The states relating to the thrust mode are dependent on the design of the subsystem. For the moment 3 states are assumed, the first is pre-heating the chamber, the second is the actual firing and third is a measurement mode to create data for analysis.

System Mode - Safe

Phase(s)	Nominal and Extended Operations
Main Capability	Securing the satellite after a failure or
Duration	NA

Safe Mode is used on most S/C to secure the system in case of a failure or a condition outside the normal mission envelope is detected. For example the Rosetta probe dropped into safe mode due to faulty star sensor detections. [38]. In the case of DelFFi it has to be determined which system are critical in the recovery process and which can be deactivated. An extensive discussion can be found in section 5-7. The duration of the mode cannot determined, as it is dependent why the S/C transitions into the mode and how fast it can be recovered from a failure event. The mode is additionally required by the FDIR-01 and FDIR-05 requirements. (See 5-1-3.)

Transition to	**Activation Criteria**	**Exit Criteria**
Main	Safe Mode drop detected	Detumbling Confirmed User Commanded

The safe mode states are defined within the Failure, Detection, Isolation, Recovery (FDIR) discussion. In total 4 states have identified to be required for DelFFi.

System Mode - EOL

Phase(s)	End of Life
Main Capability	Do not interfere with other S/C
Duration	After conclusion of extended operations to re-entry

After conclusion of the extended operations the S/C is decommissioned.

Transition to	**Activation Criteria**	**Exit Criteria**
Main	User Commanded	User Commanded User Commanded

The post-mission state of the DelFFi satellites is to turn off the transmitting of telemetry to ground and any non-essential subsystem. Remaining would be the EPS, OBC and the receiver, allowing a possible reactivation at a later point of time.

4-3-6 State definition

Every mode will have a set of possible system states. The required states for each subsystem are discussed in section 4-3-5. As a state is the physical representation of a system, the set of states should describe all possible physical configurations. However this would create several dozen states taken into account the different ADCS components which are active at different times. A compromise is to define a system state as set of subsystem modes.

Subsystem Modes

For using subsystem modes to describe the system state, the first step is to define all possible subsystem modes. This shown in figure 4-11, which lists the defined modes from each

subsystem, the power consumption and source of the information. It has been chosen to use excel to collect the subsystem modes for improved sharing within the team.

Not included in the states are the different telemetry modes which can be defined on the basis of FlexTLM developed by [12]. The reason for not including is that at the current state of the project the required telemetry from each subsystem and the communication link to earth needs to be further analysed. If these modes should be implemented, it is proposed to indicate as CDHS subsystem mode.

State Definition

With the determined subsystem modes the system states for the different system modes can be defined. Figure 4-12 shows the states for the pre-mission phase. The test system mode is excluded as the physical configuration of the system can change from day to day. For the mission phase the system states are shown in 4-13, compared to Main-LEOP, the thrust states are separated into the thrust mode. Additionally depending on the sub-phase, velocity pointing can include formation flying. The singular post-mission mode is included in figure 4-13.

Subsystem Modes		Description	Power Consumption [mW]			Note	Source
			Voltage	Orbit Avg.	Max		
FIPEX	Standby	Active SU between measurements	5	5	100		FIPEX ICD Issue 2.3
	Error	Sending Error Packages to OBC	5	5	100		FIPEX ICD Issue 2.3
	Science	Performing Measurements	5	422	1900/3000	3000 [mW]during sensor turn on (2s)	FIPEX ICD Issue 2.3
	Health Check	Testing Sensors	5	422	1900/3000	3000 [mW]during sensor turn on (2s)	FIPEX ICD Issue 2.3
µPS+	Measure	Measure relevant parameters	[TBD]	[TBD]	[TBD]	pending design	
	Pre-heating	Prepare for thruster firing	[TBD]	[TBD]	[TBD]	pending design	
	Thrusting	Executing thrust maneuver	[TBD]	[TBD]	[TBD]	pending design	
ISL	Transmitting (S-Band)_ADV	Transmitting/Receiving in S-Band + GPS	3.3	[TBD]	450		DFF-TEK-IC-1109[2.0]
	Receiving(S-Band)_ADV	Receiving in S-Band + GPS	3.3	[TBD]	150		DFF-TEK-IC-1109[2.0]
	Transmitting(UHF)_COMP	Transmitting/Receiving in UHF/VHF-Band + GPS	3.3	[TBD]	1200		DFF-TEK-IC-1109[2.0]
	Transmitting (VHF)_COMP	Receiving in UHF/VHF-Band + GPS	3.3	[TBD]	150		DFF-TEK-IC-1109[2.0]
	Storage	Deactivated Power supply to S/C	NA		60	uA - Current consumption	DFF-GOM-IC-1133[8.0]
EPS	EPS board	Active power supply to S/C	[TBD]		125		DFF-GOM-IC-1133[8.0]
	Battery board (active heater)		[TBD]	[TBD]	[TBD]		
	Battery board (no heater)		[TBD]	[TBD]	[TBD]		
ADCS	Standby	No actuation, measurements only	[TBD]	[TBD]	[TBD]	pending design	
	Detumble	Reduction of angular spin rates	[TBD]	[TBD]	[TBD]	pending design	
	Velocity Pointing + FF	Keeping attitude and execution of FF algorithm	[TBD]	[TBD]	[TBD]	pending design	
	Velocity Pointing	Keeping attitude	[TBD]	[TBD]	[TBD]	pending design	
	Thrust Control	Support thrust maneuvers	[TBD]	[TBD]	[TBD]	pending design	
COMMS	Receiver	Receive only (UHF/VHF)	[TBD]	204	204		DFF-TUD-BU-1113[4.0]
	Transmitter	Transceiver (UHF/VHF)	[TBD]	[TBD]	1655		DFF-TUD-BU-1113[4.0]
CDHS	Active OBC	Executing on-board Software	[TBD]	182	182		DFF-TUD-BU-1113[4.0]
	Boot	Initializing Software	[TBD]	182	182		
DRB	Deployment SP	Deployment of Solar Panels	[TBD]	[TBD]	[TBD]		
	Deployment AntS	Deployment of Antennas	[TBD]	NA	23		
	Standby		[TBD]	23	23		DFF-TUD-BU-1113[4.0]
Ants	Deployment Antennas	Deployment of Antennas	3.3	NA	1848	per element for 3s	DFF-ISS-IC-1144[2.0]
	Standby		3.3	28			28 DFF-ISS-IC-1144[2.0]
Ground	Testing	Testing of the different components	NA				
	Operations	Operating the S/C in flight					
	Standby	Having a coffee					

Figure 4-11: Subsystem Modes

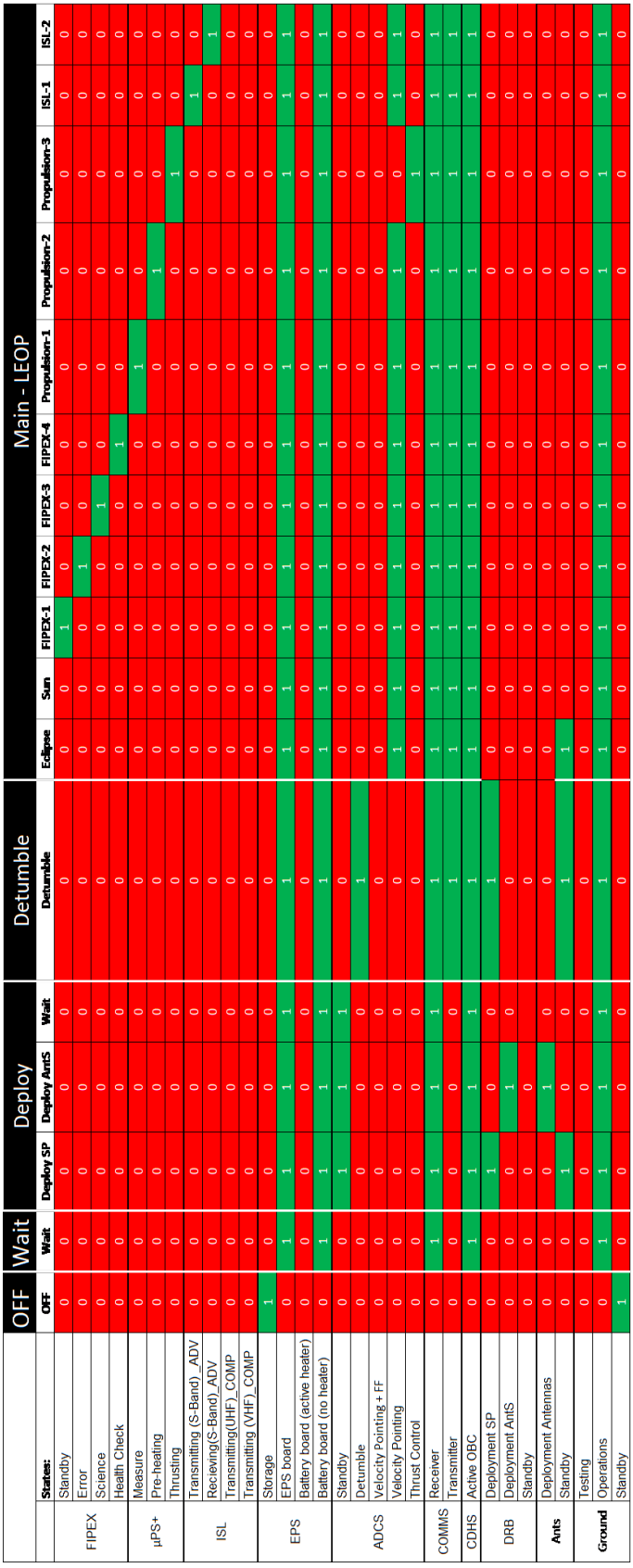


Figure 4-12: System States Pre Mission Phase

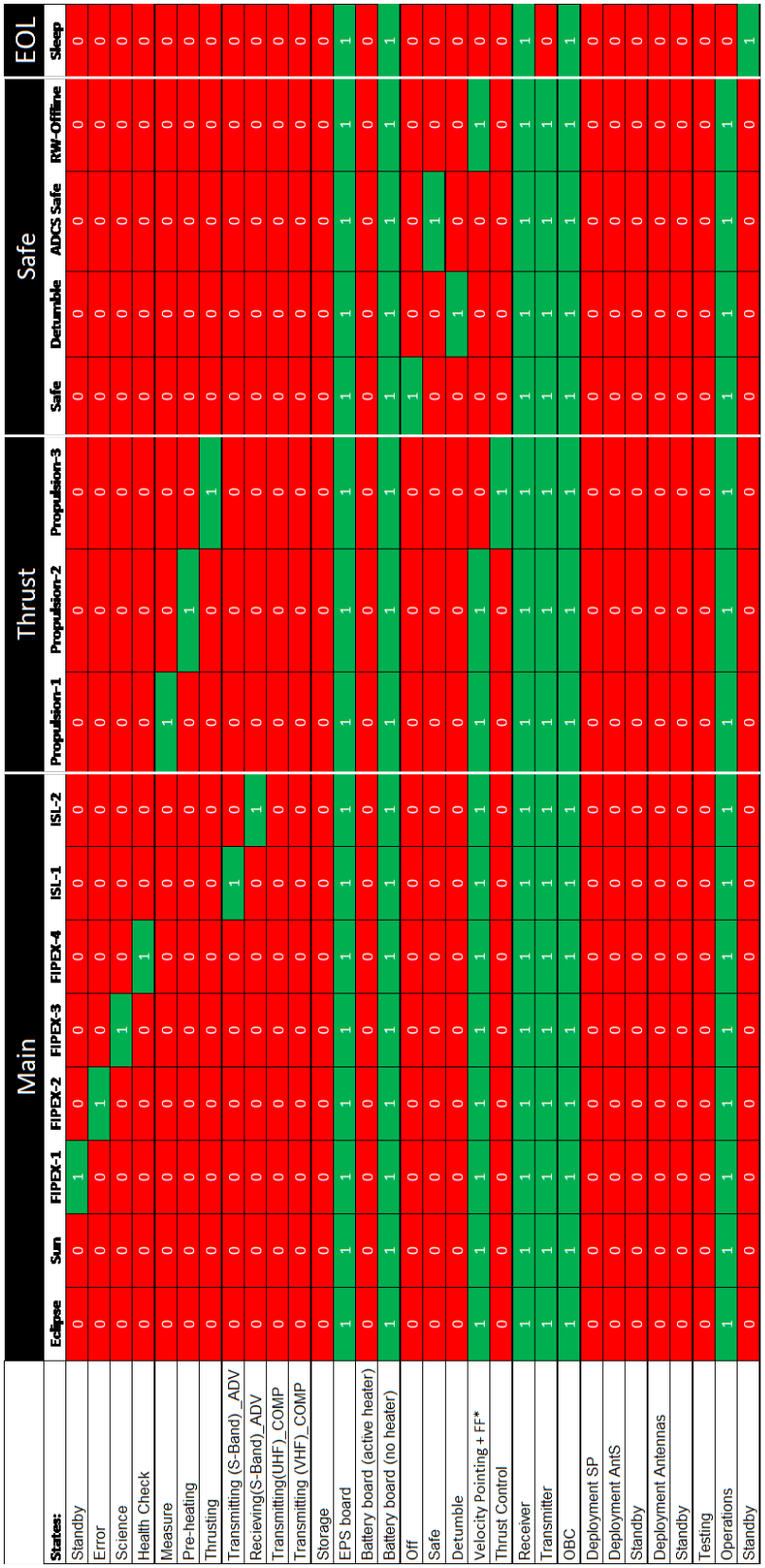


Figure 4-13: System States Mission and Post Mission Phase

4-4 Conclusion and Recommendation

The summarized research question of this chapter is how can the system modes and states be defined for the system architecture of a CubeSat CDHS?

A survey into industry standards and scientific literature showed that no dedicated methodology for defining system modes is available for CubeSats. The research performed within this chapter develops a top-down methodology for CubeSats from a general approach discussed in literature. In total 6 steps are required to define the system modes and states of a CubeSat..

1. **Operational Stakeholders:** Explore project stakeholders and determine which have an operational impact.
2. **System Phases:** Define and analyse System Phases from the mission objectives.
3. **Use Cases:** Identify use cases of the operational stakeholders.
4. **Mode Identification:** Assign use cases to system phases and combine shared outcome and capabilities to System Mode.
5. **Mode Analysis and Transition:** Analyse system modes and define activation and exit criteria for mode transition.
6. **System States and Subsystem Modes:** Define System States by determining subsystem modes and assigning them to system modes.

For step 1 the concept of operational stakeholders has been developed as part of the research. A S/C has many stakeholders, but after launch only a few are using the system. Thus for a more effective analysis, it is important to limit it to the stakeholders who are influencing system during its operation.

A second concept introduced is to define system modes as the summary of subsystem modes. The definitions provided by literature define a system state to be the list of state variables. By using subsystem modes to define a state, it allows other subsystem to develop the hardware configuration for the required use-case independent from the CDHS. This is especially for university developments, with a high turn-over in engineers/students working on the project, of advantage as it decouples some of the work while still allowing capture the operations of the spacecraft as a whole.

For CubeSats the process might seem complex as a set of modes, can be defined on the basis of experience from prior missions. However the approach used, includes analysis which can be used to define the concept of operations and by thorough analysis can remove the risk of overlooking modes or phases. This can lead to a more structured approach in developing the software architecture.

The results of application of the methodology is mission specific. For example a mission with a singular objective will have a reduced set of modes and might chose for a bottom-up approach. From academic point of view the methodology used has been applied to DelFFi only. For improved verification the methodology should be applied to CubeSats projects and compare if the used approach is applicable and usable by other engineers. A challenge is

the uniqueness of each space mission, which will lead to different choices in the development process of the operational concept. Validation of the approach is possible by implementing the proposed phases, modes and states into the software of the DelFFi satellites.

Concluding the engineering task of defining the phases, states and modes of DelFFi has been achieved by using the developed methodology. In total 10 system modes have been defined: Test, Off, Wait, Deploy, Detumble, LEOP-Main, Main, Safe, Thrust and EOL. These modes are supported by 33 system states.

For the future it is recommended to iterate over the different steps and further develop the diagrams used to improve the clarity of presentation. The methodology should be applied to different projects in the past and compare results with the developed methodology to decision made by the teams. Another recommendation would be to discuss the topic with experienced systems engineers and include their feedback. Final recommendation would be to apply the methodology to large space projects and identify what changes have to be made to have a general validity of the approach for space systems.

Chapter 5

Failure, Detection, Isolation and Recovery Concept for the DelFFi CDHS

In chapter 4 a safe mode is identified on the basis of the DelFFi use cases. The main use and characteristic of this mode is to transition the spacecraft into a configuration which reduces the impact of a failure event. Thus the Spacecraft (S/C) is required to detect when a transition from the main mode and execution of the science objectives to the safe mode is vital. From a research perspective the questions to be answered in this chapter are what is the general implementation of Failure, Detection, Isolation, Recovery (FDIR) procedures in the space industry, how can failure events be identified and how can FDIR procedures be implemented in Delft Formation Flying Experiment (DelFFi).

A discussion into the general implementation of FDIR procedures can be found in section 5-1.Failures are events during the operations of the S/C which can reduce the capabilities of the system. Methodlogies to determine and map possible failure events are discussed in section 5-2-5. The implementation of FDIR procedures is discussed in section 5-3 to 5-7. Section 5-9, provides conclusions to the research questions and recommendations.

5-1 Failure, Detection, Isolation, Recovery Overview

For answering the research question of this chapter the first step is to investigate how FDIR concepts have been implemented in past S/C. For implementing a FDIR it is important to know what failures of the subsystem are possible. Different methods for the failure analysis of a system are presented in section 5-2-5 and a method is selected and applied to the DelFFi Command and Data Handling System (CDHS).

5-1-1 FDIR Concepts as used in Industry

The FDIR ,according to [39], is implemented in large S/C by monitoring, using dedicated microcontroller, variables and verifying that they are within their defined thresholds. Depending on the severity and sensitivity of these thresholds a failure event can be detected, if a threshold is exceeded or multiple times. After detection the failure event is isolated by activating redundancy system or transferring the S/C into a safe mode. Depending on the severity of the failure and the required autonomy the S/C can recover autonomously.

According to [40] FDIR begins with the detection of failures by constant monitoring of the system and unit-level. This includes similar as the description in [39] checks if parameters received via On-Board Telemetry (OBTM) are within the predefined ranges and if transitions between modes were executed correctly. After a failure is detected there are two possibilities for isolation and recovery. The first is to turn off the affected subsystem and start using the redundant system. The second is to transfer the S/C into a safe mode/state.

As an implemented FDIR system can save a mission from a critical failure event research has been performed for S/C. There are two studies facilitated by ESA, Advanced FDIR[41] and Smart FDIR[42].

Advanced FDIR uses for the detection of failure events Kalman filtering and residual analysis to predict the condition of the spacecraft in the future. Furthermore the study proposes the use of parity-vector methods and voting techniques to compare the data from different sensors. Finally on-board simulations are used to compute expected values. The next step is the analysis of the failure, this done by Bayesian networks and model-based diagnosis with causal networks. With identification of the failure the S/C is brought into a safe configuration to isolate the failure and no further harm the system and operations. Traditionally this is done by using a safe mode, which assures survival but can prohibit the fulfilment of mission objectives. The study proposes that the FDIR transitions the S/C into a state which isolates the failure but allows to keep the maximum amount of capabilities. This requires that the FDIR knows all possible configurations and can transition autonomously between the modes and states. [41] The second study, Smart FDIR [42], mainly discusses the application of artificial intelligence in the form of models. These models are used to identify the failure and optimize the transition from a failure state to a safe configuration.

5-1-2 FDIR for CubeSats

The discussion in section 5-1-1 focusses on the implementation of FDIR procedures in large S/C. Compared to CubeSats, such as the DelFFi satellites, the CDHS consists of multiple processors with dedicated task, of which one is used for FDIR. With the higher processing power and significant man power to develop software it is necessary to adapt the FDIR procedures, as described in [39, 40, 41, 42], for CubeSats.

A CubeSat mission which aims to demonstrate novel CDHS implementations for large satellites is ESA's OPS-Sat. This mission requires the CubeSat to emulate the CDHS of large spacecraft and has accordingly increased processing and power resources, 500 [Mhz] and peak power of 30 [W]. With the mission objective to test different novel On-Board Software (OBSW) concepts, independent FDIR capabilities from the Main Computing Unit / Microcontroller (MCU) are required to return the S/C to a safe state. [43] this shows that

a traditional FDIR can be implemented into a CubeSat. On the other hand for university projects the development resources are limited and adaptions have to be made. In the past CubeSat FDIR procedures have been implemented in the development process by including redundancies and software which is able to switch between different components.

By investigating the different failure events of DelFFi a systematic implementation approach can be developed. In previous Delfi missions FDIR characteristics and functions, such as no single point of failure in Delfi C3 or the custom Delft Standard System Bus (DSSB) for the Inter-Integrated Circuit (I2C) bus in Delfi-n3xt, have been applied. The aim of this chapter is to develop FDIR functionalities and procedures for the current design of the DelFFi CDHS. The flow-down of development activities is shown in figure 5-1.

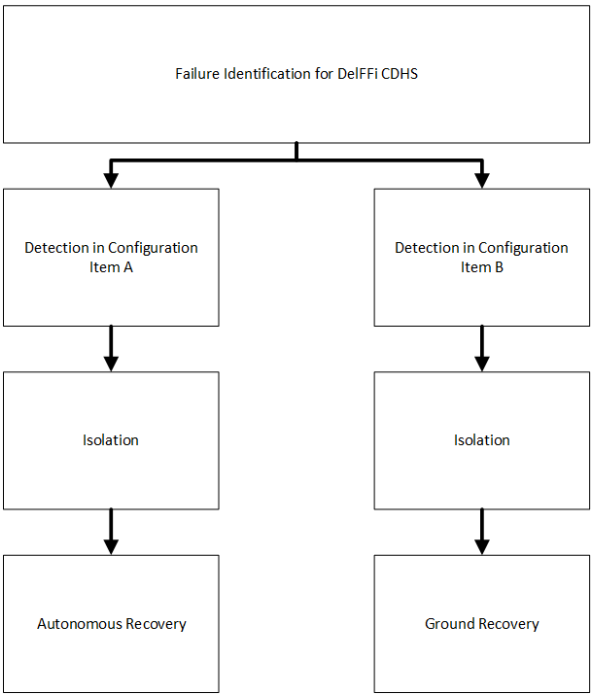


Figure 5-1: Development Methodology for the DelFFi CDHS

5-1-3 Requirements related to the FDIR for DelFFi

The capabilities of the DelFFi FDIR procedures are defined by the DelFFi system requirements. Note the requirement document [3] does not provide unique numbers for all requirements and the missing position in the ID are denoted with X. For the subsequent discussion in this chapter the requirements are assigned an unique ID.

DFF-SAT.2.6-X-XX: The CDHS shall monitor all critical subsystem parameters and have the capability to switch all subsystems to safe mode.

FDIR-01 This requirement defines that CDHS needs to be able to react to faults/failures in the critical subsystems. Critical subsystems are assumed to be CDHS, TRXUV, ADCS and the Electronic Power System (EPS). The Safe mode is not defined within the requirements.

DFF-SAT.2.6-X-XX: The CDHS shall have various degrees of autonomy.

FDIR-02 This requirement defines that both S/C should be to operate autonomously. This requirement is applicable to the CDHS as whole, but does not further define the degrees of autonomy.

DFF-SAT.2.6-X-XX: The CDHS shall implement a failure recovery approach for invalid or missed TCs.

FDIR-03 The CDHS should be able to verify if a command uploaded to the S/C is valid and transferred completely. It is assumed that valid ommand should be executable and should not lead to a critical failure event.

DFF-SAT.2.6-X-XX: CDHS shall be capable of error detection and rejection of corrupted data

FDIR-04 This requirement is mainly related to detecting errors or faults in the data. It is assumed that all data (housekeeping, payload, software and variables) is specified.

DFF-SAT.2.6-X-XX: The CDHS shall have a fault-tolerance mode that can be set by TC.

FDIR-05 The CDHS should support a safe-mode which can be set via telecommands.

DFF-SAT.2.6-X-XX: The CDHS shall attempt to detect, in selected processes, faulty processing results.

FDIR-06 The selected processes are not defined, making the requirement ambigous. For the discussion the requirment is asssumed to be related general error detection and the processes related to the execution of the OBSW and incoming OBTM.

DFF-SAT.2.6-X-XX: The OBSW and mission support software shall protect itself against infinite loops, computational errors and possible lock ups.

FDIR-07 This requirement is formulated too broad and requires specification of events which have to be handled for OBSW and mission support software separately. For the discussion it is assumed that the requirement defines failure events should be handled by the CDHS.

5-2 Failure

The first step in the development of a FDIR is the identification of possible failure events. In this section first the technical risks are identified, 5-2-1, followed by a discussion on the lessons learnt from Delfi n3xt and Delfi C3, subsection 5-2-3 and a discussion on the methodology used for identifying failures in subsection 5-2-5. The selected methodology is applied to DelFFi in 5-2-6.

5-2-1 Risk Assessment

Before investigating the failure events it is important to identify and quantify the riskof a system. During the early phases of the project a system-level analysis has been performed [44] which identified the programmtic risks of the DelFFi program.

A template for risk management of the different DelFFi subsystem has been developed by S. Kuijk. The main conclusion, in terms of programmatic risk, for the CDHS is that the current development status of the software is low and major efforts have to be made in the development. A more detailed overview of the current development status can be found in chapter 3 and the risk assessment in appendix D. The assessment at the current state mainly discusses development risk and will require updates with identified technical risks in the future. The basis for determining the technical risks can be the discussion in this chapter and lessons learnt from prior Delfi missions.

Furthermore the design choice of having a non-redundant On-board Computer (OBC) implies a increase risk. Flight heritage of the MCU and the need to use Commercial off the Shelf (COTS) components required to accept the risk. A more detailed discussion about the OBC can be found in section 5-3-1.

From past experiences, see section 5-2-3,the I2C data bus is less reliable compared to the MCU. Contrary to the failure of the OBC, I2C related risks can partly be mitigated by using I2C buffer, powercycling of the subsystems and fine-tuning the system during the test campaign. The following section, 5-2-3, identifies general failure sources from previous missions and the space environment.

5-2-2 Space Environment

The environment in Low Earth Orbit (LEO) poses unique challenges to a system which differ from the surface of the Earth. European Space Agency (ESA) defines 4 main environmental effects: Radiation, Plasma interaction, Micro-Particle and atmospheric effects. [45] Effects of Plasma, Micro-Particle and atmospheric effects are outside of the scope of this document. The major influence on the CDHS are radiation effects. An umbrella category are Single Event Effect (SEE) of which two are further explained below. During the lifetime the Total Ionization Dose (TID) is increasing and can lead to degraded performance or failures if it exceeds the limit of a component. [46]

Single Event Upset (SEU) is also known as bit-flip. It is the effect when a charged particle hits a components and changes its state. On a S/C this can result that a memory location

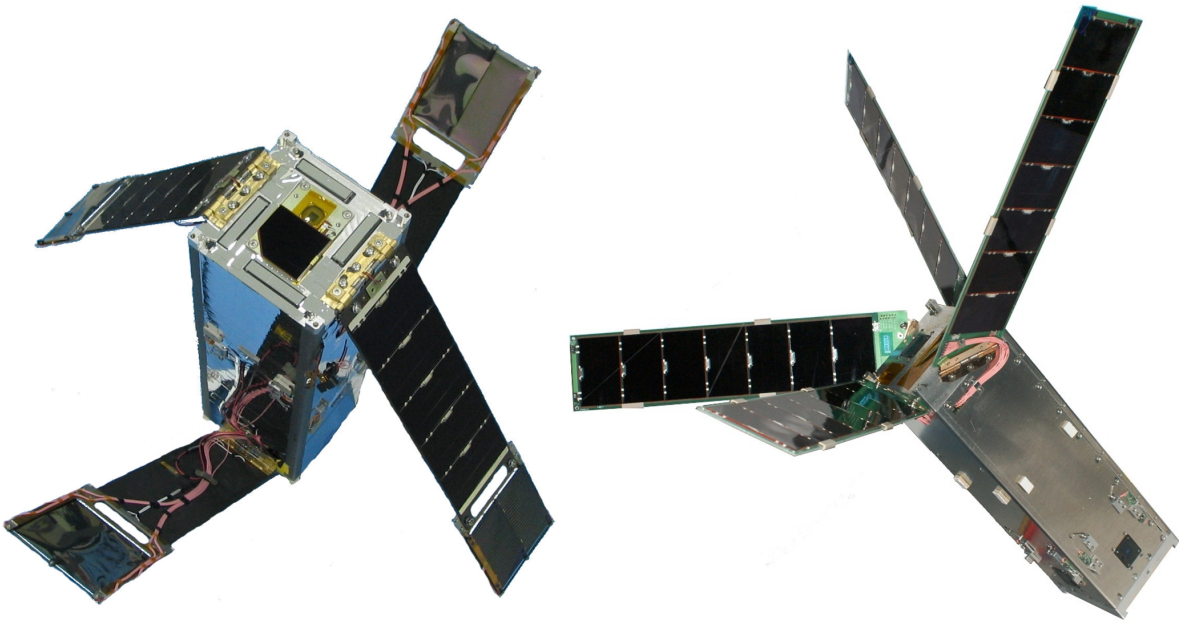


Figure 5-2: Delfi C3 (left) and Delfi - n3Xt (right) (adapted from [2])

changes it value from 0 to 1 or vice versa. During the transfer of data using the on-board data buses errors can be introduced.

Single Event Latchup (SEL) is an event in which radiation introduces a high current. It can damage the electronics of a S/C or result in locked data bus.

5-2-3 Lesson Learnt

The DelFFi CDHS is based on the flight proven heritage from Delfi n3Xt and Delfi C3. A first step is to investigate the lessons learnt in the development and operations of the predecessors. In the past decade many more CubeSats have been launched and operated. However reporting of failures in CubeSat is rare, as a CubeSat project is used to improve the reputation of a university. Currently unpublished research tries to investigate common failure causes of a satellites. Both satellites are shown in 5-2.

Delfi C3

The lessons learned from DelFi-C3 are discussed in the internal project document [47]. For the CDHS these are shortly summarized in the following discussion. The microcontroller which has been implemented on the different subsystems had issues with programming and verification of its software. The development also showed that a custom Printed Circuit Board (PCB) for the OBC has the possibility to reduce costs and compatibility issues compared to the use of the COTS FM-430 board.

The lessons learned from the use of the I2C are that for a reliable data connection, in terms of bit-error-rate, the clock speed should be at maximum 10 percent of the slowest micro-controller implemented in the system. Additionally the idle times between transmissions on the bus should chosen such that the micro-controller on the subsystems can detect their address on the I2C bus. In the transmission of data between the different subsystems parity bits have been included, however post launch analysis showed that this decreased the reliability of the system. Other issues in the I2C interface are the capacitance of the nodes, impedance of unpowered nodes and possible crosstalk.

The CDHS of Delfi-C3 included an backup mode in case the OBC fails during the mission. However, the development of this mode used significant amount of time resources which could have been spent on increasing the reliability of system in general by ,for example, including a redundant OBC. The report included suggestions for the operations and frame generation. In the telemetry it should be assured that error states can be detected without confusing with real data. Another suggestion is to include a time stamp in the telemetry and not rely on a boot counter. The last advice given in the report is that all states of the system should be defined and not limited to the activity flows.

Delfi n3Xt

Lessons learned CDHS of Delfi n3Xt are reported in [12]. For the operations of the satellite during Launch and Early Operation Phase (LEOP) a mission control room was used in which specialist could monitor the performance of the different subsystems on-board. After the LEOP the satellite has been controlled by radio operators in shifts.

One of the issues encountered in the operations of the satellite has been a regular reboot of the OBC. From the received telemetry no root cause could be determined. A problem has been that insufficient information about I2C recovery was included in the telemetry. The suggested cause of the reboot is a possible lock-up of the I2C bus.

For preventing infinity loops switch counters are included in the Delfi-N3XT code were included. For the S-band transceiver it was observed while telemetry indicated that the switch counter had the value 0, it had to be set to 0 by ground before it turned on again. The suggested cause is an error in the software or a bit flip, however in general it can be said that the switches worked to prevent infinity loops.

A proposal for DelFFi is to include the possibility to update the software. This could help in case of errors in software and algorithms in the different subsystem is detected. Hence it is suggested that is possible to update parameters of the subsystem software, such as the pressure range of the propulsion subsystem.

5-2-4 CubeSat Failure Cases

Research on the reliability of CubeSats by [48] showed that the failure of a CubeSat CDHS can directly be related to 7% of CubeSat failures. Furthermore 45 % of the CubeSats in the study are considered to be failed as no contact could be established. There are multiple causes, it can be assumed that CDHS failures are contributing to the percentage.

Within [49] the reliability of Data Buses and Power Buses are analysed from previous missions. The data has been collected using a questionnaire sent to various missions. The main conclusion for the reliability of the I2C databus is that due to the connection of many nodes and long-wiring distances bus lock-ups and performance, bit-error rate and data rate, occur. These failure cases are mitigated using bus-lockup protection, watchdog circuitry and layering of data buses.

5-2-5 Methodologies

There are quantitative and qualitative approaches used within the space sector to map and assess failure events. Within this section Fault-Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA) and Configuration Tree Analysis (CTA) are discussed. The selected methods are not a complete list but have been selected from advised methods from National Aeronautics and Space Administration (NASA) and ESA. CTA is not a formal method defined by a agency or industry, but is defined for the use within this project.

Qualitative is the evaluation of system using inherent characteristics of a system. In the context of failure analysis, this means failures are analysed by the required events leading to a failure, without specifying the probability of these events.

Quantitative methods are used if failure cases need to be evaluated in terms of their probability to occur. This can be used to determine which failure cases are realistic and need mitigation and which are deemed unlikely to occur during the mission time.

Fault Tree Analysis

The method of Fault-Tree Analysis has been developed in the early sixties of the last century and has been used in organizations such as NASA. It is a technique in which an undesirable event is specified and the system is analysed to identify all realistic path for this result to occur. A general fault events for DelFFi is shown in 5-3.

It is possible to evaluate a fault tree qualitative and quantitative. As shown in the figure a tree consists of events and gates. An event is an undesirable outcome and a gate expresses the relation between events and lead to a higher level event. It can be expressed in logic operators such as "and" and "or" It is possible to use both qualitative, by identifying the single points of failures, and quantitative, by defining the probabilities of an event to occur.[50].

If a fault-tree is to be used for a CubeSat or spacecraft in general the first step is to create a qualitative model flowing from a high level undesirable outcome to component level. This can than be used to identify the critical points of the design, which can be improved or need to be monitored by the detection part of the FDIR. In general the most unwanted failure event of a space system is the loss of signal. For example DelFi n3Xt could not be contacted or operated after the signal was lost and the mission is considered completed.[51]

For CubSats a fault-tree analysis has been implemented on DelFi-C3. The internal document [52] showed that it is possible to implement a fault-tree for CubeSat and provides suggestions

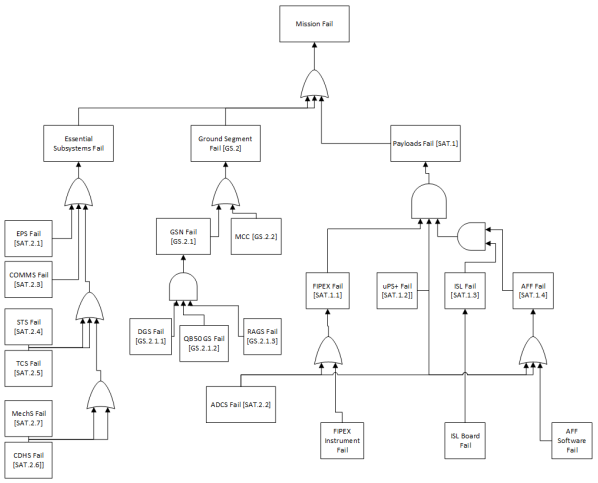


Figure 5-3: DelFFi System level Fault-Tree

and recommendations to construct such a tree. Furthermore the authors recommend to start creating a fault-tree in the conceptual design phase and assign the task to the lead system engineer of the project. For a complete fault-tree a input from all subsystems is needed and a workload of 2 man-weeks is expected to create such a model with sufficient documentations and prior knowledge.

Failure Mode and Effect (Criticality) Analysis

A different method is Failure Mode and Effect Analysis which was developed in the 1960s. [50]. Opposite to the failure-tree method it is developed from the use cases of the system. The first step is to define the system and its corresponding use-cases, following a process as discussed in section 4-3-3. For each function or use case all possible failure modes have to be identified and the consequences of such a failure evaluated. This method would have the advantage of using the prior developed use cases and expand on their analysis.

Depending on the analysis of each failure mode a severity rating is assigned. Furthermore each failure mode is investigated using root-cause analysis to find all causes which can lead to the failure mode. Consequently each failure mode is analysed on the frequency of its occurrence. This shows similar to fault-tree the process is both qualitative, by defining the failures modes on the basis of use cases, and quantitative by determining the different ratings.

Continuing for each failure cause the current control has to be identified. Control defines how a failure mode is currently mitigated and rated of how good it is at detecting the failure event. The rankings can be combined into the risk priority number (severity x frequency x detection) and criticality number (severity x frequency). This can be used to identify parts of the system which need to be improved in the design process or mitigation measures implemented as part of the FDIR procedures. An example by NASA is shown in table 5-4. [53]

The difficulty in applying FMEA to a project is that requires the full investigation of all possible failure modes and all possible failure causes. In comparison FTA only specifies to identify realistic events. A challenge in FMEA is the assignment of rankings, these should be on the basis of a quantitative analysis and evaluated using tools such as Analytical Hierarchy

Mode of Failure	Cause of Failure	Effect of Failure	Frequency (1-10)	Severity (1-10)	Chance of Detection (1-10)	Risk Priority	Design Action	Design Validation
Data Loss	Timing Error	Major Loss of data	2	9	8	144	Increase hold times 10%	Verify in timing runs
	Bit Error	Minor Loss of data	4	1	10	40	Use CRC checks to validate data	Verify in design review
Incorrect Data	SEU	Minor Loss of data	2	2	7	28	Use latch back registers	Verify in design review
	Stuck Bit	Minor Loss of data	1	2	2	4	Use CRC checks to validate data	Verify in design review
	Timing Error	Major Loss of data	2	7	8	112	Increase hold times 10%	Verify in timing runs
Chip Hang	Wrong State entered	Loss of all data	1	10	10	100	Insure all unused states have an exit mode	Verify in design review

Figure 5-4: Failure Mode Effect Analysis Example (from [53])

Process to reduce the subjectivity of the system designer. For a CubeSat project the work required to create a full FMEA has been found by [54] to be outside of the scope of a university project.

Configuration Tree Analysis

A different option is the use of a configuration tree analysis. This approach has been developed as the effort for both FTA and FMEA is estimated to be outside of the scope of this document. Compared to FTA and FMEA the focus is not on the events leading to a catastrophic failure of the S/C but on possible failures of the different configuration items of a system. For every configuration item possible failure cases are analysed and investigated. This has advantage that failures are identified which will not necessarily lead to a catastrophic failure but decrease the performance of the satellites.

With the differentiation between the configuration items team member can focus a local FDIR procedure implementation. An example of such a tree for DelFFi is shown in 5-5. It is important to note that failures assigned to configuration items are not quantified for their probability of occurrence, but only qualitative assessed if it is possible to mitigate them using a FDIR.

Conclusions

The discussion on the methodology shows that FTA and FMEA are commonly used within the aerospace industry. In general FMEA is the more complex, as it requires understanding of critical and non-critical failure modes and the assignment of weights on the basis of analysis. FTA focusses on determining which failure events can lead to a Loss of Signal (LOS) or Loss of Mission (LOM). It has the advantage that it can be used qualitatively and can be augmented with quantitative analysis. According to documentation of Delfi C3 the creation of FTA for a system requires at least 2 weeks with sufficient documentation.

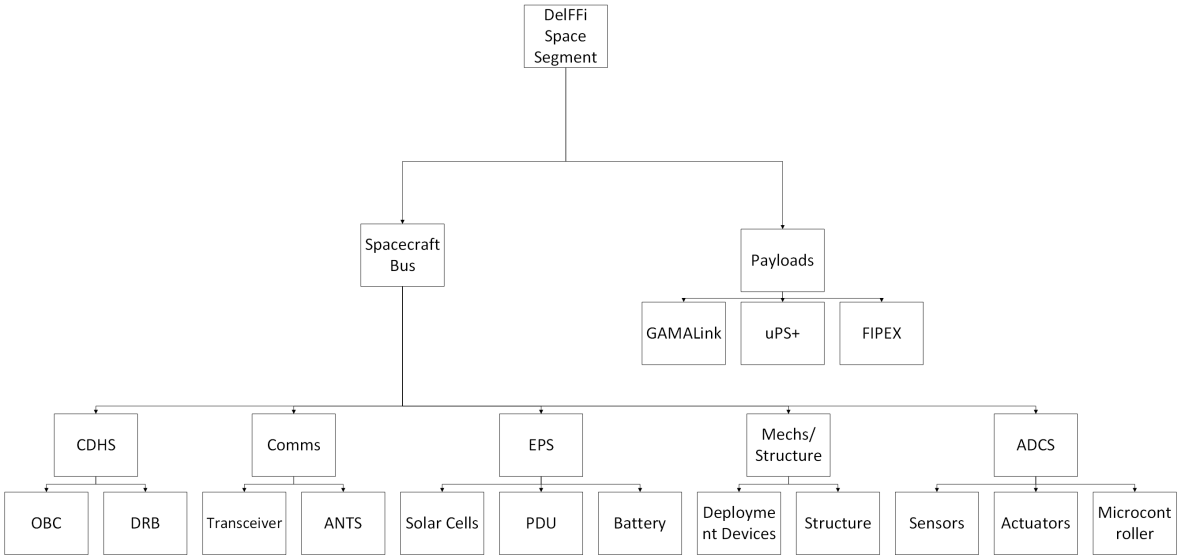


Figure 5-5: DelFFi Configuration Tree, excluding ground segment

Both FMEA and FTA focus on a system level wide analysis of the S/C. In comparison CTA only assigns failure events to configuration items. The disadvantage of the method is that it does not map the relation between different events and does not define the probability of occurrence.

For the further discussion CTA is used to determine FDIR procedures. The main argument is that the time required to perform a full analysis using FTA and FMEA is outside of the scope of this document. From an information content perspective both FTA and FMEA are superior.

5-2-6 DelFFi CDHS Failure Analysis

Within this document only the configuration item relating to the CDHS are considered, which are shown in figure 5-6. Items which are shaded grey in the figure are not considered further in the development of the FDIR procedures.

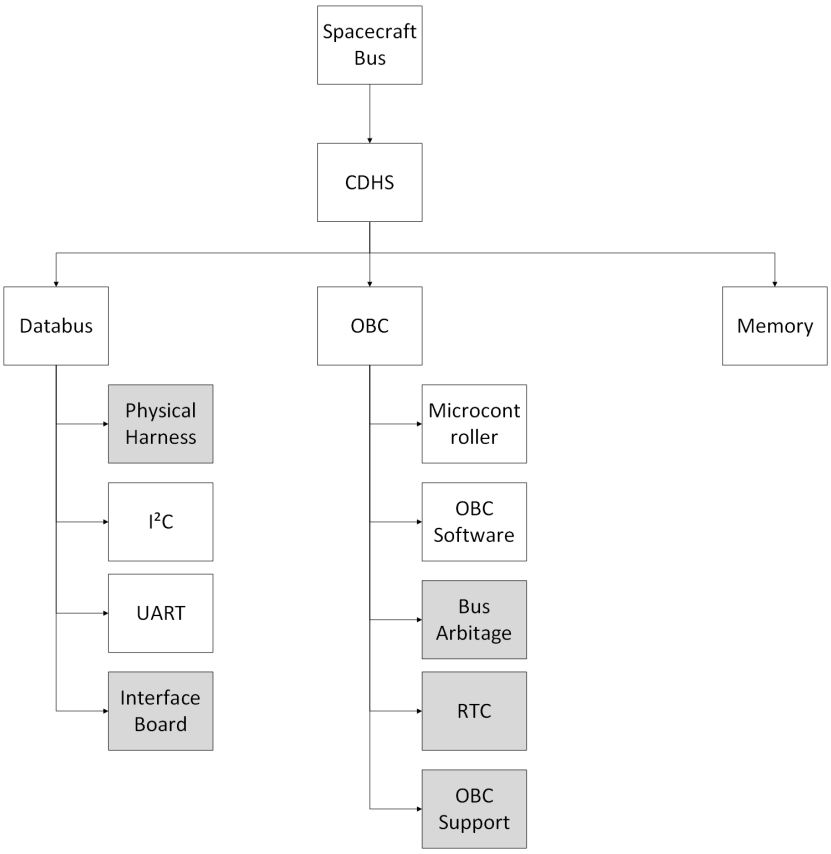


Figure 5-6: DelFFi Configuration Tree for lower level CDHS (Input adapted from [3])

5-3 Main Computing Unit / Microcontroller

This section discusses possible FDIR implementation for failure events of the MCU on the basis of the requirements: FDIR-06 and FDIR-07.

5-3-1 Failure

Two identical MCU (MSP430F2418) are implemented on the OBC and the Daughter Board (DRB). The Random Access Memory (RAM) and Flash memory of both MCU can experience changes in memory locations due to SEU. Further failure events can occur due to design flaws on the PCB and inherent to the MCU.

For the MSP430F2418, the microcontroller used on DelFFi, the radiation tolerance has been tested by a former team member. According to [8] the MSP430 series is performing up to a TID of 37 [kRad], allowing a minimum orbit life time of 5 years. Delfi-C3, using a similar MCU, continues to operate 7 years after launch. Thus with the performed radiation testing and flight heritage of Delfi-C3 failure event due to radiation are not likely to occur during the lifetime of the S/C.

In case of a radiation event, further discussed in section 5-3-1, there are three possible effects on the RAM and Flash memory of the MCU. The first is a change in value of a memory location, for example the deployment variable is changed from to 0. The impact of this event depends on the affected variable affected. A second possibility is a change in the programming of the OBSW. If such an event occurs on the Flash memory, the OBSW is permanently altered and can lead to a loss of mission. Finally, a change in an unused memory cell, does not have an impact on the operations of the S/C.

5-3-2 Detection

A change in the OBSW can result in a locked MCU by entering a or getting stuck. As a consequence the EPS watchdog timer expires, as it receives no I2C communication and starts a general power cycle of the system. In case of a changed variable or programming in the RAM the reboot will resolve failure event. This can be detected by ground by an increase of the boot counter parameter. In case of a change in the flash memory, a failure event can lead to a LOS as the faulty programming is continuously loaded into the RAM. If telemetry is received this failure event can be detected by a constantly increasing boot counter.

FDIR-06 and FDIR-07 require the CDHS to be able to detect faulty processing results and protect itself against infinity loops, computational errors and lock ups. Both requirements are difficult to meet within the current architecture of the CDHS. For detecting faulty processes and computational errors a voting logic using independent processor needs to be implemented. Thus both requirements need to be modified for the future development process.

5-3-3 Isolation and Recovery

A critical failure event on the MCU, due to radiation or a flaw of the design, cannot be isolated as no redundancy is implemented and the OBSW cannot be updated in-situ. Thus

implementation of a watchdog timer for OBC would not allow isolation or recovery of the system. Note that an infinity loop, caused by changes in RAM, which constantly requests data from the EPS is a critical failure, as the EPS watchdog is not triggered. If testing and further analysis shows that this failure is likely to occur within the lifetime of the spacecraft, an implementation of the OBC watchdog might be required.

For changes which do not affect the instructions of the OBSW it is possible to recover the S/C using parameter update telecommands. The analysis of such an event needs to be performed by operation team on-ground.

5-4 Peripheral Memory

This section discusses failure events related to the external or peripheral memory of the CDHS. FDIR-04 requires that the CDHS is capable of handling corrupted data.

5-4-1 Failure

The payload data is stored on 3 16[GB] memory cards. Similar to RAM and Flash on the MCU radiation effects can change the value of a memory location. A fault tolerant system for the data storage on-board of DelFFi has been developed by [14]. According to the author up to 3 SEU are expected to occur every second and proposes the use of a triple redundant memory.

A layout of the system and connection to the OBC is shown in figure 5-8 and its breadboard implementation in figure 5-7 . The memory cards are connected, as shown in the figure 5-7, to the OBC using the Serial Peripheral Interface (SPI) bus.

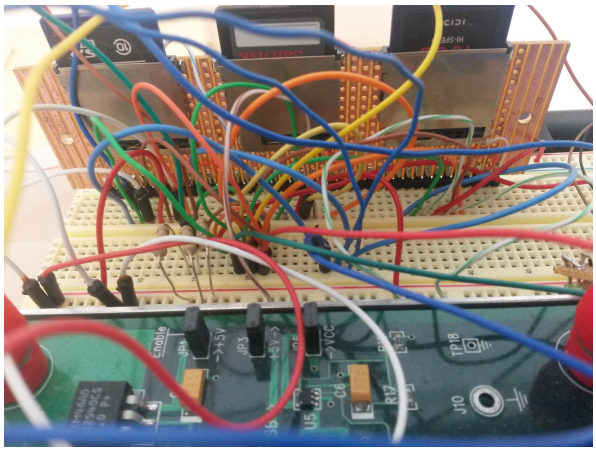


Figure 5-7: Memory Breadboard Implementation

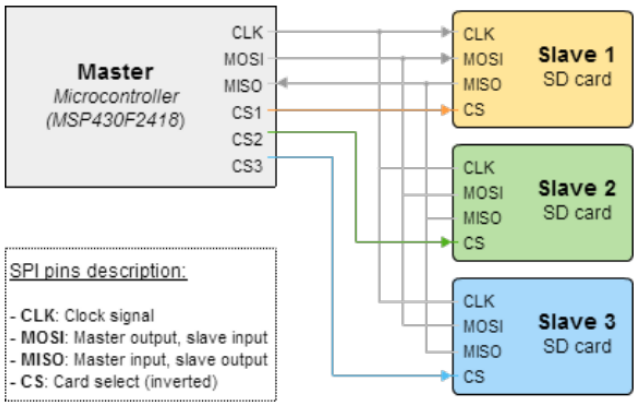


Figure 5-8: Connection Memory to OBC (From [14])

For radiation events internal error correction is used. Furthermore with the triple redundant memory cards a voting logic is implemented as shown in figure 5-9.

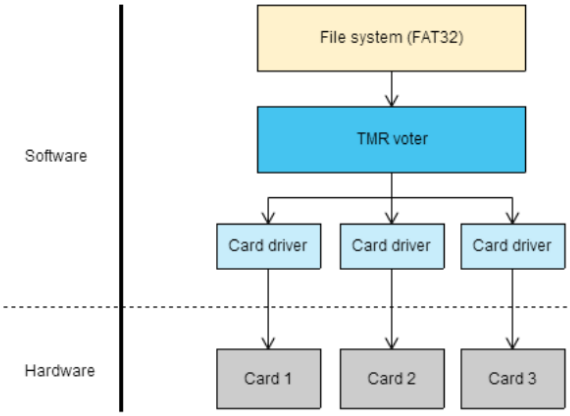


Figure 5-9: Memory Voting (From [14])

5-4-2 Detection, Isolation, and Recovery

For the development of the FDIR procedures the peripheral memory is not further considered. Following that failure handling is implemented in hardware and the service layer and that no critical information is stored in the peripheral memory. Thus possible corruption can be detected and handled on-ground. FDIR-04 requires the handling of corrupted data. A design option is to include parity bits in the intra-satellite data transmission. Based on experiences from Delfi-C3, see 5-2-3, no parity bits for have been included.

5-5 I2C

This section describes the FDIR implementation for the I2C data bus configuration item. Past experiences showed that the I2C data bus experiences lock ups during the mission. Thus the proposed implementations are part of FDIR-07. The general concept of I2C transfers is explained in section 3-2-5.

5-5-1 Failure

A detailed description of the I2C failure cases is provided in [15] and summarized in appendix B. I2C failure events can be divided into 2 categories. The first is the the effect of radiation and other subsystem on the quality of the data transferred. FDIR-04 requires the detection and rejection of corrupted data. A measure of data quality is package error rate, which can be calculated by 5-1. FDIR-04 ambiguousness can be improved by specifying the required package or bit error rate.

$$p_p = 1 - (1 - p_e)^N \tag{5-1}$$

The second category are failures, see below, which lock-down bus such that no data transmission is possible.

- R/W bit failure
- Stop Condition
- Slave misses Clock Pulse
- Low Serial Clock Line (SCL)
- Low Serial Data Line (SDA)

5-5-2 Detection

During the flight the main loop of the OBSW is executed. Between the physical and the application level the service layer is the driver for the data going through the I2C bus. A graphical representation is shown in 5-10.

Using the knowledge of failure events of the I2C bus, detection procedures can be implemented. Additionally I2C failure events can be divided into general three major rubrics: Received Not Acknowledged (NACK), SDA low and SCL low.

From the identified critical failure events Read/Write bit is resolved by the master being able to recognize an NACK, which is the case for the OBC used for DelFFi. The stop condition is resolved by performing the following I2C operation. However the slave missing the clock pulse can lead to a locked bus and remains a critical failure case. It is detected by monitoring the SDA line which remains at logic 0. The other critical failures of the I2C bus are detected similarly with monitoring the SCL and SDA. An overview of the failures and the corresponding detection is shown in table 5-1.

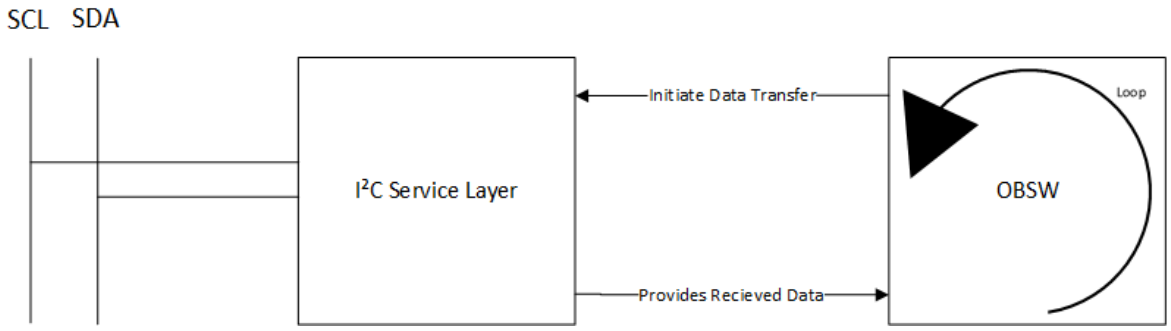


Figure 5-10: I2C Data Flow

Table 5-1: I2C Detection

Failure Event	Detected by	Note
Start condition failure	Received NACK	
Slave Request	Received NACK	
	Received ACK	Mitigated by 3-bit hamming distance
Read/Write bit	Received NACK	
Slave Acknowledge	Cyclic Redudancy Check (CRC)	
Data byte	CRC	
Stop condition	NA	Resolved by following I2C operation, possible data loss
Slave misses clock pulse	SDA monitoring	
SDA low	SDA monitoring	
SCL Low	SCL monitoring	

The implementation of the I2C FDIR component is shown in figure 5-11. As shown in the figure the first block contains the addressing of a subsystem. In case the subsystem acknowledges the master, the transfer commences and all variables are reset. In case a NACK is received the CDHS sets I2C_ error_ flag for the transmission. This flag can be in co-operated in the telemetry. If the NACK case is not applicable, the CDHS has to check if the SDA is pulled low and in sent 9 clock pulses. For the cases of a low SCL and SDA it is proposed to include bus_ lockup_ flag in the telemetry and abort the execution of any orbital manoeuvres as long the flag is active. Furthermore lock_ up_ counter has been introduced, which if exceeded a value which multiple times of the EPS watchdog, will transfer the spacecraft into the safe mode. Safe mode transitions are further discussed in section 5-7. The implementations of flags relating to the I2C has been a recommendation of the lessons learnt from Delfi n3Xt, as documented in 5-2-3 and [12].

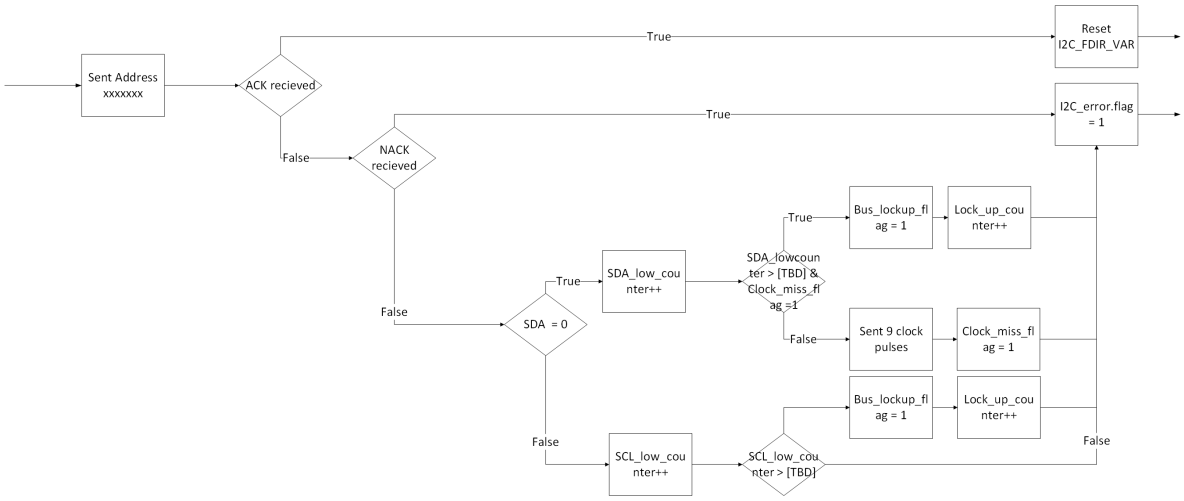


Figure 5-11: I2C Detection Activity Flow implementation

5-5-3 Isolation

The majority of the subsystems is connected to the OBC via the I2C interfaces. In the case of a locked bus no communication is possible. On the EPS a I2C watchdog timer is implemented, which power cycles the S/C if "no meaningful I2C communication" is received. [55]

From a mission objective and operations perspective it is important to communicate to ground that the execution of the formation flying commands is not possible. The current planned implementation of the formation flying is shown in figure 5-12. At this stage of the project this risk could be mitigated by implementing a timer on the uPS, which would allow the autonomous execution of the thruster.

5-5-4 Recovery

For a locked-up bus the satellite will recover autonomously due to power-cycle initiated by the watchdog timer on the EPS. The time limit before confirmed lock-up needs to be determined during testing. After the power-cycle the OBC will reboot and resume nominal operations.

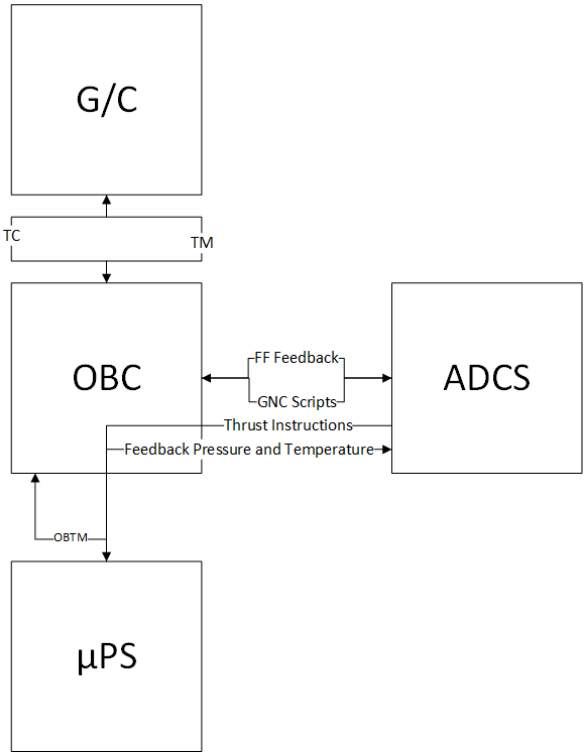


Figure 5-12: DelFFi Formation Flying Data Flows

If lock-ups continue to occur the spacecraft will transition into the Safe mode. If formation flying manoeuvres are planned during lock-up, ground control needs to upload new scripts or autonomous execution on Attitude Determination and Control System (ADCS) and the propulsion system needs to be implemented.

5-6 Universal Asynchronous Receiver/Transmitter (UART)/Flux-(Phi)-Probe-Experiment (FIPEX) and Interface

This section discusses the implementation of FDIR procedures for the UART connection to the Science Unit (SU) The discussion follows the error handling procedure as defined by the producer of the SU. The discussion in this section is related to requirement FDIR-01 and FDIR-02.

5-6-1 Failure

Failure events of the UART interface are not considered critical as a failure of the connection will allow continued operation of the S/C.

5-6-2 Detection

The failure detection for the FIPEX payload and the UART interface are treated combined. Similar to I2C the communication, as shown in figure 5-13 is handled by the UART service layer.

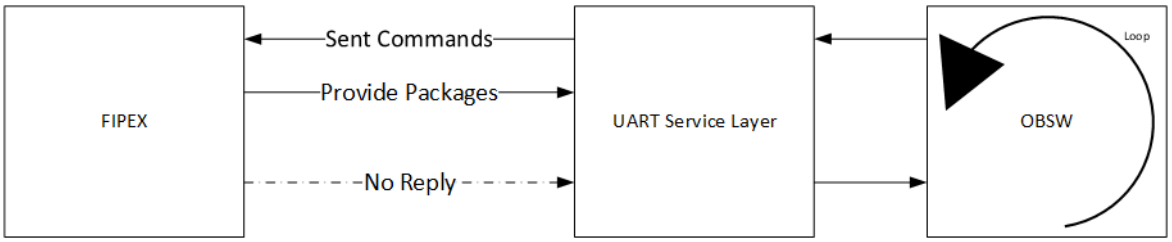


Figure 5-13: UART Data Flow

For the use of commands scripts the error handling is defined within the activity flows. However for the FDIR procedures it is proposed to create an encapsulated error handling on the basis of [6]. There are two cases to be considered, reception of an error indication from the payload and no response from the science unit. A possible implementation is shown in 5-14.

5-6-3 Isolation and Recovery

The isolation of failure events is defined by [6] as turning off the SU. The isolation process is shown in the ERROR block in the activity flow as shown in figure 5-14. As a consequence no further handling of a failure event is needed on-board and the failure of the SU is reported via the telemetry to ground. Upon reception of the telemetry ground is in charge of the recovery by analysing the telemetry received prior to the Error message. If no recovery strategy can be developed the system developer can be contacted.

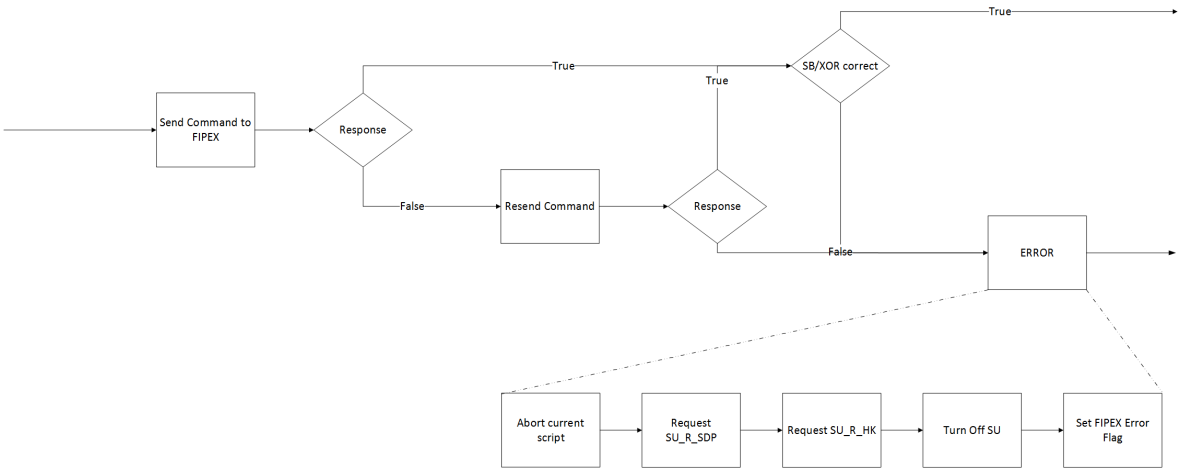


Figure 5-14: UART Detection and Isolation Activity Flow Implementation

5-7 On-Board Software

The discussion in this section, relating to the on-board software, is related to requirements FDIR-01, FDIR-03, FDIR-04, FDIR-05 and FDIR-06. While prior sections discussed FDIR software implementations for I2C and UART, a focus in this section is on software flaws, Telecommand (TC), OBTM and downlinked telemetry. In this section failure event which occur on other subsystem are linked to the FDIR procedures of the CDHS.

5-7-1 Failure

Criticality it defined by effects the failure sources have on the execution of the software and state of the satellite. Software bugs which would lead to a general failure of the CDHS are considered to be detectable during testing and mitigated during the development of the OBSW. However, absence of software flaws cannot be assumed in the flight firmware. [56] A detailed discussion on failure events is provided in the detection section.

5-7-2 Detection

The detection of failure events can be divided into the four previously mentioned categories, which are discussed separately.

Telecommands

FDIR-03 prohibits the execution of invalid or sent erroneous commands. The CDHS of DelFFi accepts 5 kinds of TC of which updates to volatile and non-volatile parameters andI2C pass-through commands are important for the development of FDIR procedures. Critical subsystems for the operations of the S/C are the EPS, CDHS and the TRXUV. Their corresponding I2C addresses are provided in 5-2. Missed TC can be detected by ground by analysing the received Telemetry (TM).

A list of prohibited commands and parameters is defined in the following paragraphs. Invalid commands or wrongly addressed, are non-critical, resulting in a time-out of the I2C bus or the transmission of NACK from the addressed subsystem.

Table 5-2: I2C Addresses of critical Subsystems

Subsystem		Address	
	dec	bin	hex
EPS	17	0010001	11
OBC	15	0001111	F
TRXUV	34	0100010	22

EPS With the EPS it is possible to control the power supply of the different subsystem. Depending on the operations payloads such as the SU or propulsion system might be turned on or off. However the three critical subsystems should not be allowed to be turned off at

any time during the mission. Note the EPS is powered constantly by the solar arrays and the battery and OBC and TRXUV might be power-cycled in case of a timed-out EPS watchdog timer. The prohibited commands for the EPS are shown in table 5-3. The channels on are not yet defined, requiring the arguments of the commands to be defined at a later stage.

Table 5-3: Forbidden I2C commands for EPS

Name	Hex	Bytes	Note
SET_Output	00001001	1	See EPS I2C for selection of channels
SET_ SINGLE_ OUTPUT	00001010	4	Allows setting of only 1 channel

TRXUV According to [57] it is not possible to turn, using I2Ccommands, the receiving functions off. TC restriction prohibit the execution of a ground sent command. A non-critical option is to reset the micro-controller of the receiver and turn off the transmitter. This functionality is required for power cycling and at the end of life of satellite due to regulations, see section 4-3-2.

OBC Changes to the OBC are made by updating parameters in the volatile RAM or non-volatile Flash memory of the MCU. A critical value is the operational mode, in specific the wait mode which is active after deployment of the satellite. It is possible that by accident the spacecraft is returned to the wait state.

Implementation The detection of forbidden I2C commands and changing of the operational mode to "wait" is shown in 5-15. For I2C pass-through commands the first check is if the EPS is addressed, followed by a check if the commands are forbidden. In case an error flag for the EPS will be set, such that ground control can recognize the error in the uploaded commands and adapt it in future uploads. For the OBC a check for the wait mode is included, and if triggered an error flag is set.

On-Board Telemetry

The OBC can only detect failure events of the S/C as part of the incoming data from other subsystems. A failure in one of the other subsystem can require a system wide response of the CDHS. An overview of which transmitted parameters can indicate a failure is shown in table 5-4. Note that only subsystems are mentioned which have telemetry fields which can be used for the detection of a failure. Furthermore, a failure event can be detected if a subsystem is not replying to queries from the OBC.

The failure detection with the on-board telemetry is proposed to be implemented within the collection of the telemetry for downlink. The implementation is shown in figure 5-16.

Software Bugs

The majority of Bugs should be detected during the ground testing of the satellite. Thus all remaining bugs of the software should not lead to a critical failure of the system. Furthermore any detection of such a bug needs to be performed on-ground on the basis of received telemetry.

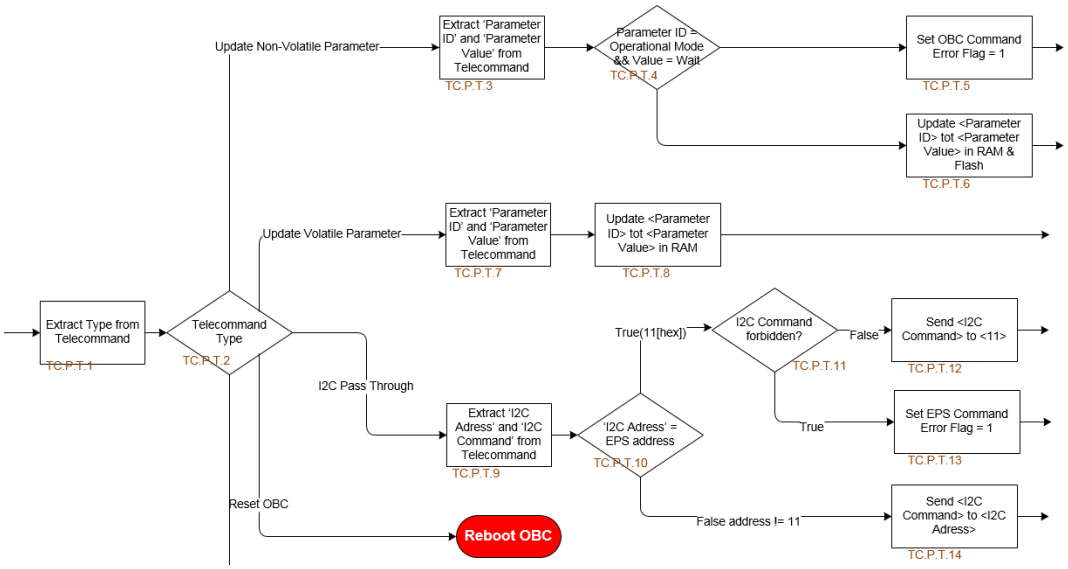


Figure 5-15: Telecommand Failure Detection

Table 5-4: OBTM Failure Detection Parameters

Subsystem	Parameter	Description
EPS	$T_{battery}$	Temperature of the battery
EPS	I_{switch}	Current, can be used to detect over-current and power consumption by system
ADCS	Subsystem Mode	Safe Mode / No RW / Detumble
GAMALink	Error Code	Error Code specified by ICD [58]
uPS	Subsystem Mode = Error	Error Code provided by Subsystem

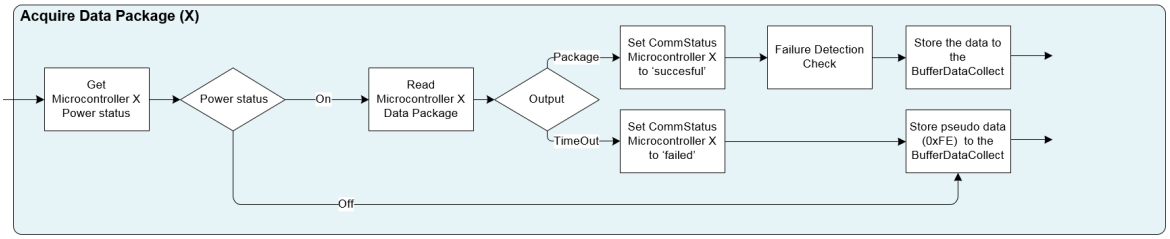


Figure 5-16: OBTM Failure Detection

Table 5-5: Error Flags in the telemetry

Bit Location	Flag	Section
1	Bus_ Lock_ Up	3-2-5
2	I2C_ error	3-2-5
3-10	Lock_ up _ counter	3-2-5 / 5-7
11	FIPEX_ ERROR	5-6
12	OBC_ Error	5-7
13	EPS_ Command	5-7
14	T_ Warn	5-7
15	FDIR_ Override	5-7

Downlinked Telemetry

During the operation of the satellite it is possible that failure events are detected due to the downlinked telemetry. These error might be known from testing of the Engineering Model (EM) and Flight Model (FM) or require the analysis of the operation team. For the scope of this discussion it is assumed that detection occurs on-ground. In the defintion of the FDIR procedures error flags have been introduced into the activity flows of the OBSW. These flags are proposed to be downlinked as part of the normal telemetry. An overview of the flags is shown in table 5-5.

The flags are added to the Flextlm package OBC_ Err as shown in figure 5-17. Currently there are 7 flags defined with each can have the value true, hence logic one and false, logic 0. This means for the currently defined flags 7 bits are required. Furthermore a lockup counter for the I2C data interface is included, with 8 bit assigned, allowing a maximum value of 255. As future developments might add more flags 49 bit are kept as a reserve.

The OBC_ Err package is currently not part of the general downlink. However as the size of 64 bit is only 3% of the maximum package size in [12] it is assumed that it can be added to downlinked telemetry.

							Value			Desired update frequency [Hz]			Average total bit size [1/s]		
Subsystem	Packet group	FlexTLM packet	Description	Parameter	Bit size	Unit	Minimum	Maximum	Transfer function	Minimum	Normal	Maximum	Minimum	Normal	Maximum
OBC	OBC basic parameters	OBC_BASIC	ETC Time Tag		32	s	0	429496729.5	[value] · 0.1	1	1	1	32.00	32.00	32.00
			Boot counter		16	-	0	65535	[value]	1	1	1	16.00	16.00	16.00
			Frame counter		31	-	0	2.15E+09	[value]	1	1	1	31.00	31.00	31.00
			OBC temperature 1		5	°C				1	1	1	1.00	1.00	1.00
			OBC temperature 2		5	°C				1	1	1	1.00	1.00	1.00
	OBC reserved parameters	OBC_RESERVED	Reserved for switch counters or similar		50					1	1	1	50.00	50.00	50.00
	Whole Orbit Data (WOD)	OBC_WOD	Mode		1					1	1	1	1.00	1.00	1.00
			Battery bus voltage		8	V				1	1	1	8.00	8.00	8.00
			Battery bus current		8	A				1	1	1	8.00	8.00	8.00
			3V3 bus current		8	A				1	1	1	8.00	8.00	8.00
			5V bus current		8	A				1	1	1	8.00	8.00	8.00
			Temperature COMMS		8	°C				1	1	1	8.00	8.00	8.00
			Temperature EPS		8	°C				1	1	1	8.00	8.00	8.00
			Temperature battery		8	°C				1	1	1	8.00	8.00	8.00
	Telecommand Feedback	OBC_TCFB	Telecommand feedback		128	-				1	1	1	128	128	128
	Abnormal state information	OBC_ERR	FDIR Flags		7					1	1	1	7	7	7
			Lock_Up_Counter		8					1	1	1	8	8	8
			Reserved Space		49					1	1	1	49	49	49
Total													380	380	380

Figure 5-17: OBC FlexTLM Packages

5-7-3 Isolation

In comparison to the I2C, MCU, UART and Memory the isolation and recovery possible for failure events in the OBSW. The fist step is to investigate which subsystems are critical for the survival of the spacecraft and how much autonomy is needed. An overview for DelFFi is shown in figure 5-18.

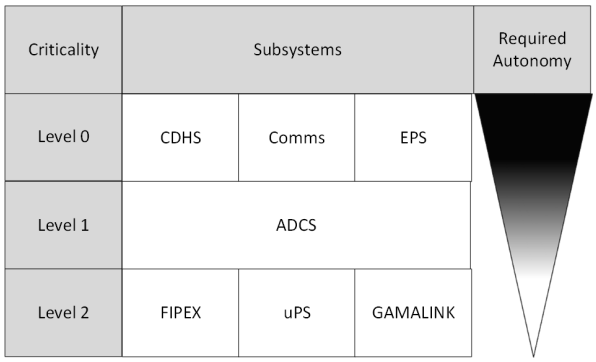


Figure 5-18: DelFFi Criticality and Autonomy Matrix

Ground Commands

For errors in the TC the isolation is combined with the detection. This is shown in figure 5-15. No further action is required.

On-Board Telemetry

Figure 5-18 shows the degree of autonomy required related to the criticality of the subsystems. A failure in the payloads (level 2) can be isolated by turning of the subsystem.

For the critical subsystems EPS provides informations which can be used in the development of FDIR procedure. The variable T_{bat} is related to the thermal control of the S/C. Prior research proposes to include a heater on the batteries. [59] If the temperature on the batteries is outside of the qualification range lasting damages to the battery can occur. A mitigation possibility is to turn on and off the battery heater using automatic control implemented on the EPS and us FDIR procedures as second safe-guard to ensure the survival of the missions. The maximum power consumption of the heater is 7.2 [W], requiring a safe state to support the power consumption. More analysis into the thermal control and power budget of the S/C is required to determine if this is necessary. Currently a warning flag has been included in the TM transferred to ground.

The I_{switch} parameters from the EPS provides the current consumption of each load channel. It can be used to monitor over-current conditions and the power consumption. A simplified layout of the EPS is shown in figure 5-19. The power, P_{cons} consumed by the spacecraft can be calculated using equation 5-2.

$$P_{cons} = \sum_{i=1}^{3} I_{switch_i} 3.3V + \sum_{i=4}^{6} I_{switch_i} 5V \tag{5-2}$$

The subsystems of the S/C are protected from over-current events using load switches implemented on the hardware level of the OBC. With the information provided by the EPS an additional software -side protection can be implemented. In case the current in one of the load-lines exceeds the limit it is power-cycled and if the error cannot be resolved turned off. This isolation procedure is implemented for subsystem criticality level 1 and 2.

By monitoring the power provided to the subsystems, non-nominal conditions can be detected by comparing to expected values. If these limits are exceeded the spacecraft is transitioned into a safe state.

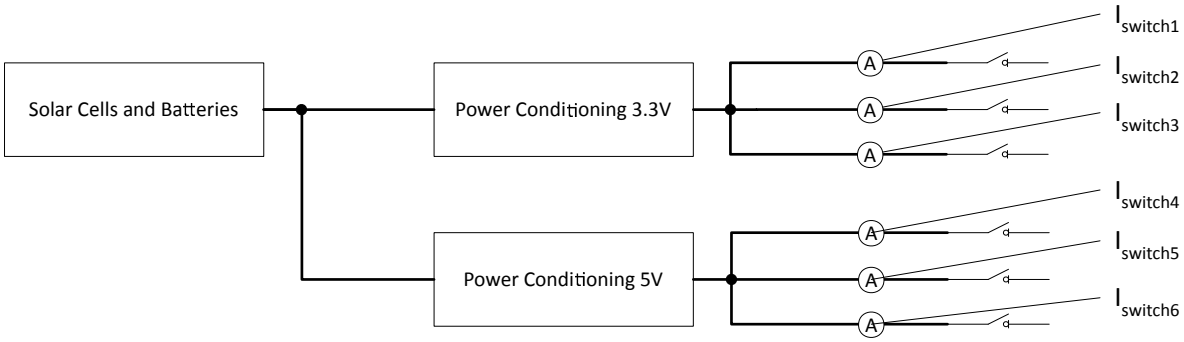


Figure 5-19: Simplified EPS layout (adapted from [55])

The ADCS is critical as the mission objectives require velocity pointing within 5 degree [3]. If this limit is exceeded the satellite needs to perform detumbling during which the operation of formation flying and FIPEX payload is not possible. It is hence proposed to defined 3 safe modes states relating to the failure states of the ADCS.

Safe Mode State Detumbling The mission objectives of Formation Flying and the operation of the SU require the S/C +Z axis to be aligned with velocity vector. [36]. The spacecraft should enter the nominal and extended mission phase in a detumbled state. However due to natural perbutations the S/C drifts from the required attitude and requires further detumbling. This is defined by the requirement (DFF-SAT-2.2-X-XX) below:

DFF-SAT-2.2-X-XX: Detumbling mode is activated when the S/C rate is above 1Âž/s. The ADCS shall use rate estimates and actuators to reduce the rate below 1Âž/s.

Safe Mode State Safe Mode ADCS A a safe mode has been defined as subsystem mode for the ADCS. In case this mode is active no attitude actuation is performed and the the attitude of the satellites will begin to drift. This means the operation of payloads is not possible.

System Safe Mode State RW For formation flying the use of Reaction Wheels (RW) is required. It is not possible to counter the torque created by misalignments solely with magnetotorquers. [37]. In case a failure of the RW is detected the execution of thrust commands needs to stopped.

For error codes provided by the payloads GAMALink and the uPS, the failure is isolated by turning the subsystems off. The impact on the formation flying needs to be investigated. The error handling of the FIPEX payload is discussed in 5-6.

Software Bugs

If software bugs are detected by ground after launch of the S/C it possible by selecting a different operational mode to skip the bug. In case the bug is caused by faulty parameters these can be updated by ground with a correct value.

Downlinked Telemetry

In case failure events are detected on ground, FDIR-5 requires the S/C to able transition into a general safe state. Figure 5-18 shows that the CDHS, EPS and the Comms are considered to be required for the survival of the S/C. Thus general safe mode state is define with these subsystems are active only.

Isolation Implementation

Within this section the implementation of the FDIR procedures into the states diagram and activity flows is discussed.

Implementation State diagram The definition of the system states is shown in 5-20. Three safe system states are defined on the basis of ADCS failures. Additionally another state is defined to allow ground to move the S/C into a general safe configuration by TC or if a safe mode transition is activated if power, current and I2C lock-up limits are exceeded.

Implementation Activity Flows As shown in figure 5-16 the failures detection for OBTM is included in the collection of telemetry. Within the block: "FDIR Detection Check" the isolation of the failure event is included. An extended view of the software block is shown in figure 5-21.

5-7-4 Recovery

There two possibilities for recovery, ground based and the autonomous recovery on-board. The OBC can only detects the error states from other subsystems. An exception is safe mode state for detumbling, which does not required ground in the loop. An overview is shown in figure 5-22.

The activity flows for the safe mode, including the recovery for detumble is shown in figure 5-23. An important observation is that only the detumble safe state allows an automatic return to the main mode.

		Safe			
	States:	**Safe**	**Detumble**	**ADCS Safe**	**RW-Offline**
FIPEX	Standby	0	0	0	0
	Error	0	0	0	0
	Science	0	0	0	0
	Health Check	0	0	0	0
µPS+	Measure	0	0	0	0
	Pre-heating	0	0	0	0
	Thrusting	0	0	0	0
ISL	Transmitting (S-Band) _ADV	0	0	0	0
	Recieving(S-Band)_ADV	0	0	0	0
	Transmitting(UHF)_COMP	0	0	0	0
	Transmitting (VHF)_COMP	0	0	0	0
EPS	Storage	0	0	0	0
	EPS board	1	1	1	1
	Battery board (active heater)	0	0	0	0
	Battery board (no heater)	1	1	1	1
ADCS	Off	1	0	0	0
	Safe	0	0	1	0
	Detumble	0	1	0	0
	Velocity Pointing + FF*	0	0	0	1
	Thrust Control	0	0	0	0
COMMS	Receiver	1	1	1	1
	Transmitter	1	1	1	1
CDHS	OBC	1	1	1	1
DRB	Deployment SP	0	0	0	0
	Deployment AntS	0	0	0	0
	Standby	0	0	0	0
Ants	Deployment Antennas	0	0	0	0
	Standby	0	0	0	0
Ground	Testing	0	0	0	0
	Operations	1	1	1	1
	Standby	0	0	0	0

*** in Extended Operations without FF**

Figure 5-20: Safe Mode States

5-8 Deployment FDIR

For the deployment of the appendages three failure events have to be taken into account. The first is the failure of the local MCU, second is a stuck bus and third is missing confirmation of the deployment by the OBTM. The following sections will discuss the implementation of FDIR procedures for the deployment of the antennas in 5-8-1 and the deployment of the solar array wings in 5-8-2 using the DRB.

The deployment is treated separately as the duration of the system phase is significant shorter than Detumble, LEO, nominal and extended mission.

5-8-1 Antenna Deployment

For the antenna board, AntS, all three failures can be handled by the FDIR of the S/C. The subsystem has two redundant micro-controllers which are able to perform the deployment of the antennas. Furthermore feature of the chosen antenna system is a subsystem mode which automatically deploy the antennas.[35]. The status of the antennas is requested every second using the "Report Deployment status" I2C command. As soon as the deployment of all 4 antennas is confirmed the satellite transitions into the detumble mode.

In case no deployment is confirmed, the OBSW, will continue to request the deployment status from the antenna board. If this situation occurs deployment of the antenna could have failed or a false negative is detected. In the first case, which is detected by ground if no telemetry is received from the S/C, the operations team can issue commands to retry the

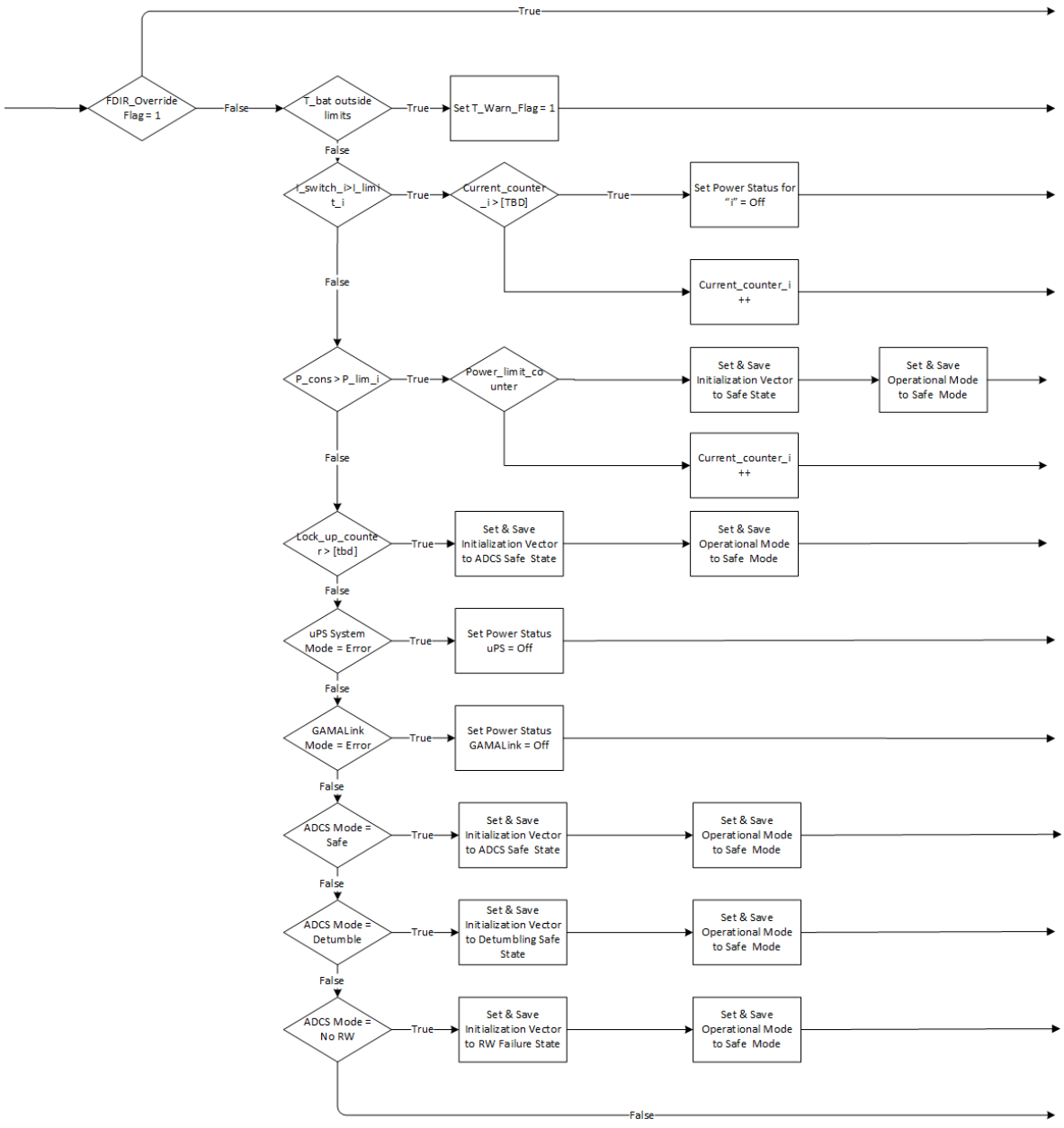


Figure 5-21: Safe Mode Isolation Activity Flows

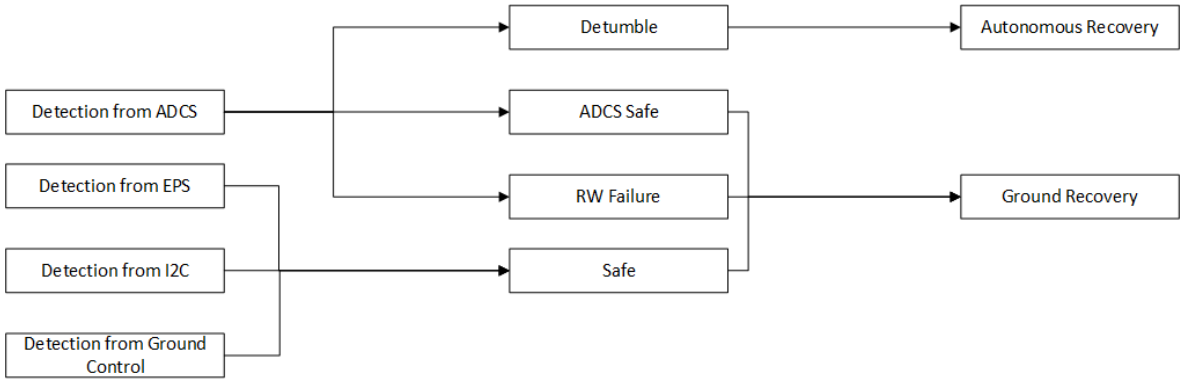


Figure 5-22: Safe Mode Recovery Overview

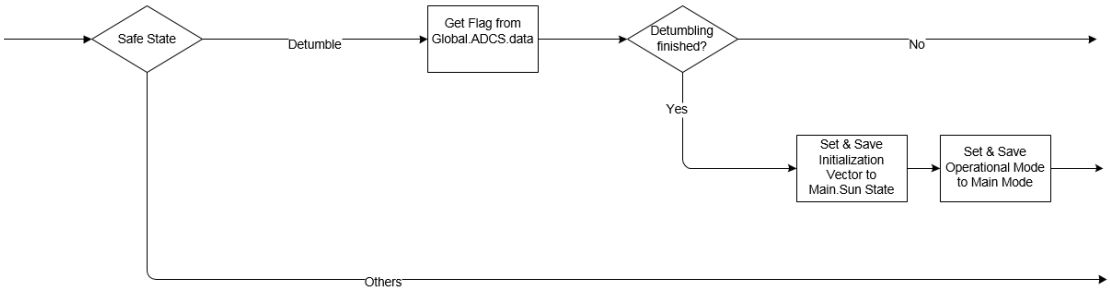


Figure 5-23: Safe Mode Activity Flows

deployment. For the second case, a time-out is implemented which transition the S/C into detumble mode. The activity flow is shown in figure 5-24.

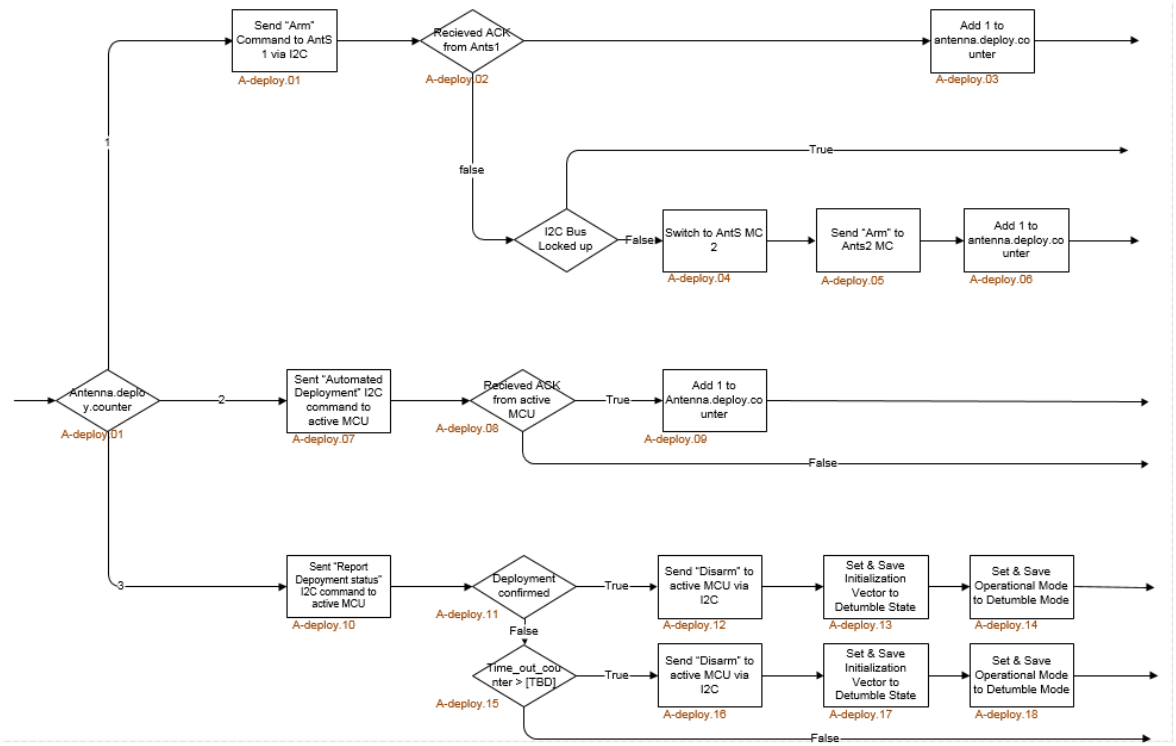


Figure 5-24: Antenna Deployment Activity Flows

5-8-2 Solar Array Deployment

The activity flows for the deployment of the solar array wings are shown in figure 5-25. Opposite to the discussion for the antenna deployment, section 5-8-1, no FDIR procedure are implemented. The first reason is that the satellite during deployment and the detumble phase is not dependent on the extra power created by the wings. Thus in case the deployment fails, or the I2C commands are not received by the DRB , these can be resent by ground at a later stage. Second reason is that the implementation of the deployment sequence on the DRB is not finally defined.

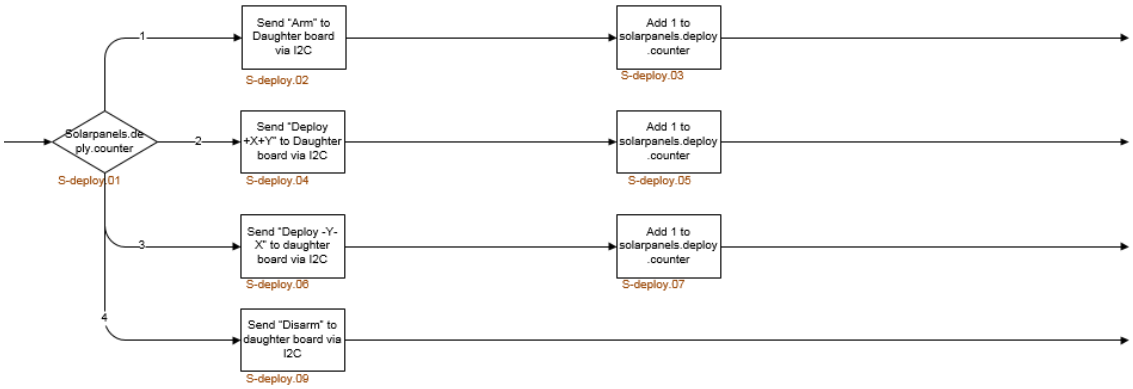


Figure 5-25: Solar Array Wings Deployment Activity Flows

5-9 Conclusions and Recommendations

The investigation in section 5-1 gave a general overview over Failure, Detection, Isolation, Recovery procedures in a S/C. It showed that FDIR procedures are implemented into large spacecrafts. Current developments, as described in [41, 42] and relevant literature, shows that the implementation on large satellites is highly complex and outside of the capabilities and resources available for the development of a CubeSat within a university setting.

For understanding which failure events can occur different methods can be applied to the S/C or a single subsystem. The standard methods within the space industry, Fault-Tree Analysis and Failure Mode and Effect Analysis, can relate single failure events and determine the probability of occurrence. However for the discussion within this document the required analysis was considered to be outside of the scope. For defining the areas where FDIR procedures can be implemented on the DelFFi CDHS the CTA method is used. The method allows to assign failure cases to the different configuration items and allows the compartmentalized development of the FDIR.

The question how can a FDIR be implemented for the DelFFi CDHS within the constraints of a CubeSat is answered in the following paragraphs. On the basis of the CTA configuration items are selected which allow for implementation. For the DelFFi CDHS these are the MCU, the external Memory, the I2C and UART data buses and the OBSW.

The MCU is, assumed on the experience from Delfi n3xt and C3, considered to be unlikely to experience a critical failure event during the mission. In case of a failure any detection, isolation and recovery has to be performed by ground control.

A triple-redundant external memory has been developed by a former student [14]. The data on the external memory is not critical to the operations of the S/C and internal error correction is implemented. Thus the configuration item is not considered critical. In case there is a failure event it has to be detected by ground and possible mitigation measures implemented.

In past mission the I2C data bus has been identified as source of failure which can lock the satellite data bus. Lessons learnt from prior missions, section 5-2-3 and research performed by [49] showed that failure events related to the I2C data bus need to be handled. The formation

flying objective of the mission requires that the thruster is activated at a certain time. In case of a locked-up bus this is not possible and needs to be reported to ground. Lessons learnt showed that more information is required about the behaviour of the I2C bus in flight. Hence flags indicating the bus status are implemented into the OBSW.For the UART bus the FDIR logic follows the procedure as defined by [6].

A major configuration item of the CDHS is the OBSW. For the discussion the item has been further divided into 4 distinct areas: Telecommands, OBTM, Software Bugs and Telemetry. TC can be detected by checking the upload for forbidden commands. Any command is considered prohibit if it would lead to a loss of signal of the S/C, such as turning off the TRXUV. The on-board telemetry provides informations of the different subsystems to the OBC, which if a failure is detected can autonomously react to it. For non-critical systems, such as the payloads, the respective subsystems are turned off. For cases which are critical to the mission objectives the spacecraft can be transferred into the safe mode. Possible software bugs need to be detected during testing of the spacecraft. The OBSW prepares the telemetry which is to be downlinked to earth. The operations team can decided if a failure event occurred on the basis of the transferred telemetry and can transition the S/C into a safe state with only OBC, TRXUV and EPS active. A recommendation from the lessons learnt of Delfi n3xt is the inclusion of boot loaders into the OBSW. With the ability to upload new software more recovery options are available for the ground team and capabilities of the satellite can be improved in-flight. However, this would require more powerful computational abilities as currently implemented on DelFFi.

Concluding the discussion showed that it is possible to systematically develop a FDIR for a CubeSat. While the discussion focusses on the CDHS, a similar method can be used for other the subsystems of a CubeSat.

The developed FDIR are partially compliant with the defined DelFFi requirements, see table 5-6. Following the initial analysis following recommendation relating to the requirements can be made. For FDIR-01 critical subsystem parameters have been defined within the discussion, yet it is inherently not possible to switch all subsystem to safe mode. All developed FDIR procedures are compliant to FDIR-02 as the requirement is defined too generally and is applicable to every system which is not operated by hand. The OBSW is partial compliant to FDIR-03, the developed FDIR prohibit the execution of TC which could lead to a critical failure. On the other hand the designed procedures do not verify if a TC is transferred completely. As a consequence the design of the OBSW is currently non-compliant and the requirement needs to be reviewed or these functions be implemented into the software architecture. The peripheral memory is compliant to FDIR-04, per contra both interfaces and the OBSW are currently non-compliant and require an analysis into the requirement. Implemented FDIR procedures are fully compliant with FDIR-05. Analysis showed that both FDIR-06 and FDIR-07 are defined too broad to be implemented into a CubeSat. As a consequence the developed FDIR procedures are non-compliant and thus require rework of both requirements

Table 5-6: DelFFi FDIR Procedures Requirements Compliancy Table

	Main Computing Unit	Peripheral Memory	I²C Interface	UART Interface + FIPEX	OBSW
FDIR-01	NA	NA	NA	C	NC/RR
FDIR-02	C	C	C	C	C
FDIR-03	NA	NA	NA	NA	NC/RR
FDIR-04	NA	C	NC/RR	NC/RR	NC/RR
FDIR-05	NA	NA	NA	NA	C
FDIR-06	NC/RR	NA	NC/RR	NA	NA
FDIR-07	NC/RR	NA	NA	NA	NA

C	**Compliant**
NC	**Non-Compliant**
NC/RR	**Non-Compliant, Requirement Review**
NA	**Not Applicable**

As recommendation results from testing of the engineering model have to be taken into account. This can include new failure events which are not treated or parts of the FDIR which are never triggered. There might also be the case that the FDIR is too sensitive and introduces new failures to the system. For future Delfi programs it is proposed to begin failure analysis of the chosen design in phase 0/A by creating a FTA. This would support the risk management and focus the developments on areas which are lacking. For a more complete understanding FMEA can be used on the basis of the determined use cases. Finally the FDIR procedures should be part of the system architecture definition from the first day of the system definition. The approach chosen in the discussion was to surgically add FDIR blocks to already defined software which can be improved by developing the system with a FDIR in mind.

Chapter 6

Conclusions

The objective of the research was to continue the development of the CDHS system architecture of DelFFi by developing a methodology to determine the system modes and implementing Failure, Detection, Isolation, Recovery procedures into the software architecture for the DelFFi Command and Data Handling System (CDHS). Within this chapter, the conclusions of the thesis work in support of the research objective are presented.

The first set of research questions discuss the current development status of DelFFi and what improvements can be made in future projects. A summary of the current development status of the different components is provided in figure 6-1.

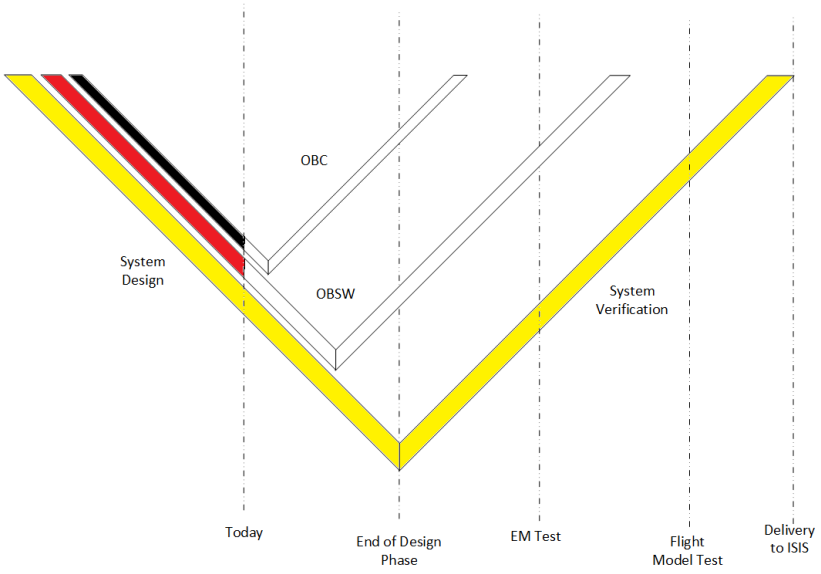


Figure 6-1: System Development Status (Nested V-diagrams adapted from [60])

The figure shows that the hardware implementation of DelFFi is currently progressing with the two main components Daughter Board (DRB) and On-board Computer (OBC) at least

in prototype stage and undergoing early testing. It hence can be concluded that the hardware for the CDHS is nearing the end of the design phase with a Technical Readiness Level (TRL) at least 4. Currently, the software of the CDHS is partly defined in the activity flows created by prior team members on the basis of Delfi - n3Xt and augmented by the research performed within this document. For the service layer of the On-Board Software (OBSW), some drivers are available which currently are used to test the OBC. Within this thesis the software architecture has been further developed, but cannot be considered to be complete. Thus, the next steps are to complete the definition of the software and begin coding. These activities will require significant resources in terms of manpower and time. Improvements for future CDHS are identified in chapter 7.

During the development of the software architecture a question within the team arose: What are the system modes of the DelFFi satellites? Within chapter 4, on the basis of relevant literature in industry standards and academia a 6-step methodology was developed:

1. **Operational Stakeholders:** Explore project stakeholders and determine which have an operational impact.
2. **System Phases:** Define and analyse System Phases from the mission objectives.
3. **Use Cases:** Identify use cases of the operational stakeholders.
4. **Mode Identification:** Assign use cases to system phases and combine shared outcome and capabilities to System Mode.
5. **Mode Analysis and Transition:** Analyse system modes and define activation and exit criteria for mode transition.
6. **System States and Subsystem Modes:** Define System States by determining subsystem modes and assigning them to system modes.

The methodology has been implemented for DelFFi and a set of 10 modes have been defined: Test, Off, Wait, Deploy, Detumble, LEOP, Main, Thrust, Safe and EOL. These modes are supported by 33 system states.

Two important concepts have been developed as part of the methodology. The first is to limit the analysis required by defining operational stakeholders. A space mission can have many different stakeholder, of which only a few have influence on the operations or use of the system in orbit. A second recommendation is to define the system modes and states no later than in phase B of the project. By phase B, hardware configuration of the spacecraft is not yet fully determined, however the use cases or functions of each required subsystem is known. Consequently, a system state in the methodology developed in this thesis is defined to be a set of subsystem modes. This allows subsystems to be developed more independently and reduce the required number of interface updates. This is especially useful for university projects where team members are fluctuating and resources for staff are limited

In conclusion the development of this methodology showed that it is possible to determine system modes using a top-down methodology. The findings of the discussion will be submitted as an abstract to a conference which will give other teams the chance to use and verify the developed methodology.

Chapter 5 developed an implementation approach for a Failure, Detection, Isolation, Recovery (FDIR) procedures for the DelFFi CDHS. Within the chapter an overview of how FDIR is implemented in large satellites is given and what the challenges are for the implementation in CubeSats. It can be concluded that due to the limited resources on a CubeSat, the concept and implementation will need to be changed compared to larger satellites. The research question addressing what FDIR procedures are, has been answered and it can be concluded that, independent of the size of the Spacecraft (S/C), the FDIR will improve the reliability of a space system.

An important part of the FDIR is to determine what failures are realistically possible. There are several different methods available of which Fault-Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA) have been investigated more closely for the use on CubeSats. Analysis showed that both methods help to map and assess possible failure events and allow the ranking of these using quantitative analysis. For discussion within this thesis, due to time limitations, both methods are considered outside of the scope of the DelFFi project and a third method, Configuration Tree Analysis (CTA), was analysed. The CTA method only assigns known failures to the different configuration items, but does not map the relation between the various failures and the paths that can lead to failure. This is advantageous for developing an FDIR for a CubeSat subsystem as it allows the division of the work to the individual configuration items.

Using CTA, five areas have been identified which are investigated further: Main Computing Unit / Microcontroller (MCU), Memory, the I2C and UART data buses and the OBSW. For every item, first, possible failure events were analysed and on the basis steps for detection, isolation and recovery were defined. In the process, the software architecture definition of the DelFFi CDHS was expanded to include error handling on the basis of requirements.

Part of the objective was to develop the FDIR of the DelFFi CDHS. In this thesis the FDIR procedures have been developed upto the definition within the activity flows. Remaining work is the actual implementation into software and creating manuals for handling on-ground. Another open point is the development of a full operations concept, which defines how non-nominal situations relating to missions objectives are dealt with.

The research within thesis this successfully developed the missing steps between the mission objectives and the system modes and states of a CubeSat. Furthermore the research shows that it is possible to develop FDIR procedures within the constraints of a CubeSats. Not only has this work progressed the DelFFi project status, but it has been identified to be beneficial for other CubeSat teams to implement as well.

Chapter 7

Recommendations

The conclusions provided in chapter 6 showed which components of the research objective have been achieved and how the research questions have been answered within this document.

A remaining research question is: What improvements can be identified for future Delfi missions or an evolution of DelFFi? This question can be answered on the basis of the discussion within this document and more general programmatic problems.

In section 1-3, reasons have been given why the thesis project had to evolve. For the future of the DelFFi program it is recommended that the planning used within the project is updated constantly, to reflect the true status of the project, and is kept realistic.

Additionally, the work within this project showed that satellite projects need specialized engineers. The systems engineering analysis of the Command and Data Handling System (CDHS) within this document can form the basis for future development. However for the actual implementation into code a software engineering student or professional should be acquired by the project.

A final point from the project perspective is that the speed of decision making needs to be improved. If decisions about designs, are faster, the amount of evolutions required in thesis work from different students decreases and the project can move at a faster pace.

From the hardware perspective several recommendations can be made. The Main Computing Unit / Microcontroller (MCU) has proven flight-history from Delfi-n3Xt and Delfi-C3. However compared to missions such as Ops-Sat the capabilities are significantly lower. Within the prior literature study the CDHS implementation of different CubeSat has been discussed and it is observed that On-Board Software (OBSW) is built on the basis of Real-Time Operating System (RTOS) and supports the use of boot-loaders. Two examples are the UWE series of satellites and ESTCube-1. An RTOS will require more resources, in terms of memory and processing power, to be implemented and hence increases the demand of power to be supplied to the CDHS. However, with a more capable MCU improvements in the development process of the OBSW can be made, as students from electrical and software engineering might be more interested to work on the project and the development of software can be more easily divided between several engineers.

A second point for recommendation is the data bus in future DelFFi satellites. All Delfi satellites use the Inter-Integrated Circuit (I2C) standard for the transfer of On-Board Telemetry (OBTM). Research by Jasper Bouwmeester showed that most CubeSats which implement I2C experienced failure events and mitigation measures needed to be taken. Within DelFFi, all subsystems, except the Science Unit (SU), are connected using one I2C bus. This means that a locked bus will prohibit any communication with other subsystems. A mitigation approach would be to layer the subsystem in different bus networks. DelFFi could be divided into three networks: EPS+TRXUV+DRB+AntS / ADCS+uPS / GAMALink. In case a failure event occurs in the bus which is not connected to Electronic Power System (EPS) the subsystem can be power cycled without relying on the EPS watchdog. Furthermore prior work by Remco Schoemaker showed that it is possible to include a Bluetooth chip on a CubeSat. A research opportunity would be to move from wired data interfaces, using I2C, to a wireless CubeSat design using Bluetooth.

The methodology developed in chapter 4 for determining System Modes and States leads to several recommendations for DelFFi and further research. From the research objective a methodology has been developed and applied to the DelFFi mission. For better verification this methodology has to be applied to different projects and performed by different persons. On the basis of this feedback the method can iterated and improved. A first step in this direction will be a possible publication and a possible exchange with a CubeSat project in Costa Rica. For DelFFi and future Delfi programs, the main recommendation is to keep the mission analysis up-to-date and use the methodology developed in this thesis to determine the system phase and modes at an earlier stage in the project.

From the discussion of the Failure, Detection, Isolation, Recovery (FDIR) several recommendations can be drawn. First is to use failure-analysis methods such as Fault-Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA) on a system level to identify critical functionalities of the system. Furthermore, the technical risk analysis performed identified the development status of the OBSW as a critical item for the project. As a consequence, a focus on the implementation of OBSW should made. The FDIR has been developed at a late stage in the project life cycle, in the the future it would be advisable to include the considerations made in chapter 5 at an earlier stage of the design process. Finally the FDIR has been implemented within architecture definition, but as the rest of the OBSW needs to be implemented as actual code and verified by testing.

With the results of the research performed within this thesis and the recommendations made in this chapter, future Delfi missions can be developed more systematic and the reliability of the satellites increased.

Appendix A

Emails

A-1 Detumble

Detumble mode time https://webmail.tudelft.nl/owa/?ae=Item&t=IPM.Note&id=RgAAAA...

Detumble mode time

morteza989@gmail.com on behalf of Morteza [m.haghayegh@student.tudelft.nl]
Sent: 10 April 2015 11:00
To: Frederik Bräuer

Dear Fred Bräuer,

Attached to this email you can find the table which shows the amount of orbits needed to detumble. For each case the timing sequence of the ADCS has been changed and the initial angular velocity represents the initial angular velocity for which the satellite can still be detumbled for that specific case. For the DelFFi mission case E has been chosen as the most feasible where we would expect an initial angular velocity of 45 deg/s in all three axis. In this case the satellite would be detumbled within 6.75 orbits.

Table 9: Overview of the cases for determining the optimal timing sequence interval

Case	Sensor	Algorithm	Actuator	Hold	Total	ω_0	nr. of orbits
A	0.1 s	~ 0 s	1.4 s	0.5 s	2 s	10 deg/s	1
B	0.1 s	~ 0 s	1 s	0.4 s	1.5 s	15 deg/s	1.25
C	0.1 s	~ 0 s	0.5 s	0.3 s	0.9 s	25 deg/s	1.75
D	0.1 s	~ 0 s	0.2 s	0.3 s	0.6 s	35 deg/s	3
E	0.1 s	~ 0 s	0.1 s	0.3 s	0.5 s	45 deg/s	6.75
F	0.1 s	~ 0 s	0.1 s	0.2 s	0.4 s	55 deg/s	6.25
G	0.1 s	~ 0 s	0.05 s	0.05 s	0.2 s	110 deg/s	11.5
H	0.05 s	~ 0 s	0.05 s	~0 s	0.1 s	190 deg/s	31

Kind regards,

Morteza Haghayegh

Space Systems Engineering - DelFFi ADCS
Aerospace Department - TU Delft
E: m.haghayegh@student.tudelft.nl
T: +31 15 278 7586

Appendix B

I2C Failure Cases

Failure	Cause	Impact
Start condition failure	Bit flip on SDA or SCL	Slave does not notice start condition and does not acknowledge the data transfer
Slave Request failure	Bitflip on SDA	Slave device does not ACK when it missed start condition
R/W bit failure	Bitflip on SDA	Master and Slave are waiting for ACK and may enter into a infinite loop
Slave acknowledge failure	Bitflip on SDA	Master reads ACK/-NACK while the slave is not present and vice-versa.
Data byte failure	Bitflip on SDA	A bitflip occurs on the ninth clock pulse which results the Master will read a sequence of ones.
Stop condition failure	Stop condition not recognized	The slave misses the stop conditions and continues writing on the bus.
Slave misses clock pulses	Bitflip on SCL	The slave and the master get of sync resulting in the SDA line being low indefinitely
Data line pulled low indefinitely	Electrical short, lockup in the controllers	SDA pulled down indefinitely
SCL pulled low indefinitely	Short, Clock stretching, latch-up in I2C controller	SCL pulled down indefinitely

Appendix C

Budgets

C-1 Power

DelFFi Power Consumption Breakdown

Specified Power Budget Mode Overview

Subsystem	S/S Mode	Nominal Power	Boot	Delay	Deploy	Detumbling			Main (Velocity Pointing Control)			Main + QB50			Formation Flying			Formation Control		
						Sun	Eclipse	Orbit	Sun	Eclipse	Orbit	Sun	Eclipse	Orbit	Sun	Eclipse	Orbit	Sun	Eclipse	Orbit
FIPEX	Standby	200 mW						0.0%			0.0%		22.5%	9.0%		22.5%	9.0%			0.0%
	Science	2000 mW						0.0%			0.0%	25.0%		15.0%	25.0%		15.0%			0.0%
µPS+	Measure	22 mW						0.0%			0.0%			0.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	Pre-heating	10000 mW						0.0%			0.0%			0.0%			0.0%	40.0%		24.0%
	Thrusting	500 mW						0.0%			0.0%			0.0%			0.0%	10.0%		6.0%
ISL	Transmitting	6600 mW						0.0%			0.0%			0.0%	5.0%		3.0%			0.0%
	Receiving	495 mW						0.0%			0.0%			0.0%	5.0%		3.0%			0.0%
	GPS positioning	50 mW						0.0%	100.0%		60.0%	100.0%		60.0%	100.0%		60.0%	100.0%		60.0%
EPS	EPS board	125 mW	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	Battery board (no heater)	0 mW	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
ADCS	Sun Sensors	204 mW						0.0%	100.0%	50.0%	80.0%	100.0%	50.0%	80.0%	100.0%	50.0%	80.0%	100.0%	50.0%	80.0%
	Magnetorquer System	600 mW				100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	20.0%	100.0%	52.0%
	Magnetorquer (Off)	72 mW	100.0%	100.0%	100.0%			0.0%			0.0%			0.0%			0.0%			0.0%
	Reaction Wheel System	800 mW						0.0%			0.0%			0.0%			0.0%	100.0%		60.0%
	Main Board ADCS1 (ARM	530 mW						0.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	Main Board ADCS2 (XME	150 mW				100.0%	100.0%	100.0%			0.0%			0.0%			0.0%			0.0%
COMMS	Receiver	204 mW	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	Transmitter	1655 mW				15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%	15.0%
CDHS	Main Board OBC	182 mW	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
MechS	Deployment Knife	1920 mW			100.0%			0.0%			0.0%			0.0%			0.0%			0.0%
	Microcontroller + Support	23 mW			100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	Thermal Sensors	1 mW			100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Total Specified Consumed Power [W]:			0.583	0.583	2.527	1.533	1.533	1.533	2.167	2.015	2.106	2.666	2.060	2.424	3.044	2.082	2.659	6.559	2.037	4.750
Total Specified Available Power [W]:			3.895	3.895	3.895	10.164	5.200	6.098	11.088	5.200	6.653	11.088	5.200	6.653	11.088	5.200	6.653	10.164	5.200	6.098
Budget surplus / deficit [W]:			3.312	3.312	1.368	8.631	3.667	4.565	8.921	3.185	4.546	8.422	3.140	4.229	8.044	3.118	3.994	3.605	3.163	1.348

Appendix D

Technical Risk Assessment: CDHS

Technical risk assessment: CDHS

Description: This document contains the risk management of CDHS of the DelFFi program

Subsystem(s) involved:	ADCS	CDHS	COMMS	EPS	MechS	STS	TCS	FIPEX	µPS+	ISL	AFF	GSE	GSN	Launch
		x			x									

Revision Record and Authorization

Issue	Date	Author / Editor	Reviewer checked	PM approved	Affected Section(s)	Description of change
1.1	10.06.2015	S van Kuijk			2	Implemented changes of tech tree
1.0	03.06.2015	Frederik Bräuer			All	Initial Release

Action Items

TBW	TBD	TBC	Applicable Section(s)	Description of action item

List of Used References

SLR code	Version	Data/Variable

Table of Contents

1 Introduction

Within this document a high-level risk assessment of the Command and Data Handling Subsystem is performed. The aim is to identify the risk in the current development status. In chapter 2 a subsystem overview is provided with the corresponding TRL for each configuration item. Chapter 3 identifies the programmatic risks of the subsystem, which are mapped in chapter 5 in a risk map. Finally chapter 6 proposes mitigation for the identified risks.

2 System description

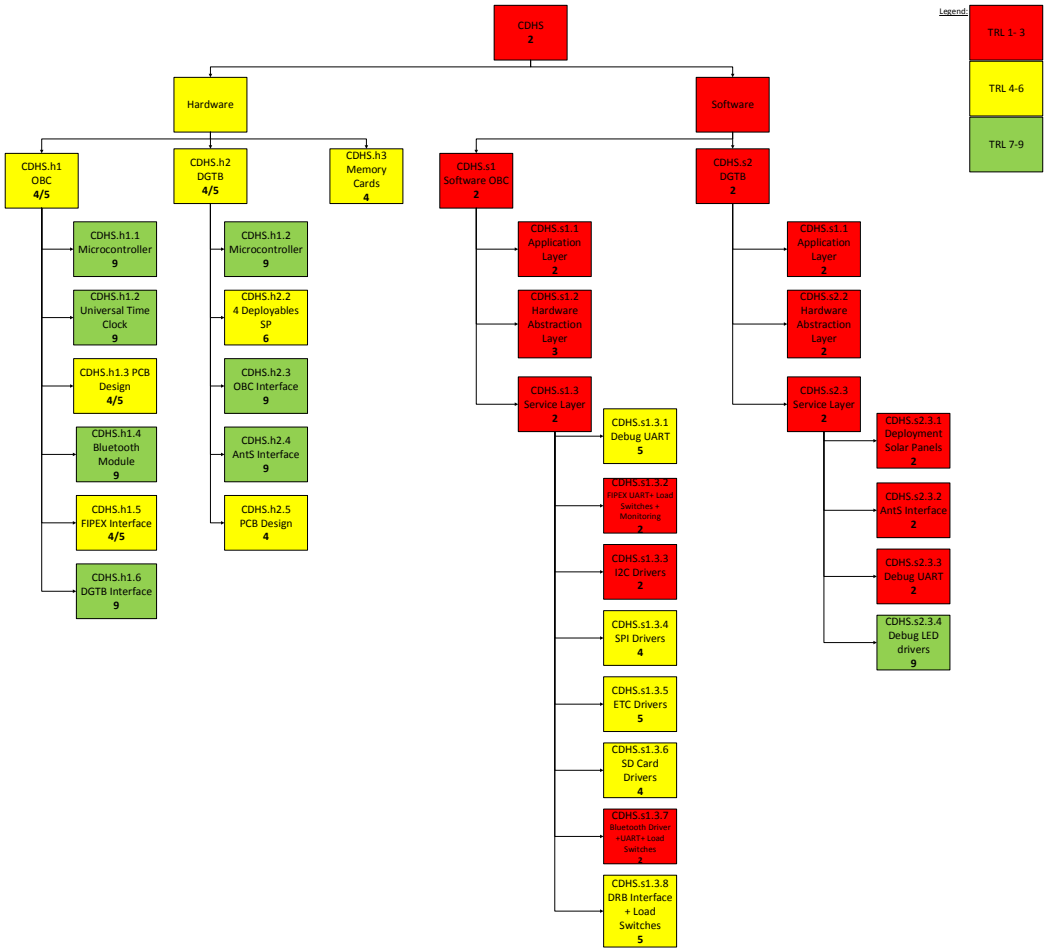


Table 2-1: Components

Hardware:		
ID	**Description**	**TRL**
CHDS.h1.	OBC	4/5
CDHS.h1.1	Microcontroller	9
CDHS.h1.2	Universal Time Clock	9
CDHS.h1.3	PCB Design	4/5
CDHS.h1.4	Bluetooth Module	9
CDHS.h1.5	FiPEX Interface	4/5
CDHS.h1.6	DGTB Interface	9
CHDS.h2.	DGTB Board	4/5
CDHS.h2.1	Microcontroller	9
CDHS.h2.2	4 Deployable solar panels	6
CDHS.h2.3	OBC Interface	9
CDHS.h2.4	Antenna Board Interface	9

CDHS.h2.5	PCB Design	4
CHDS.h3.	Memory Cards	4
Software		
ID	**Description**	**TRL**
CDHS.s1.	Software OBC	2
CDHS.s1.1	Application Layer OBC	2
CDHS.s1.2	Hardware abstraction layer	3
CDHS.s1.3	Service layer OBC	
CDHS.s1.3.1.	Debug UART	5
CDHS.s1.3.2.	FIPEX UART + Load switches +Monitoring	2
CDHS.s1.3.3.	I2C Drivers	2
CDHS.s1.3.4.	SPI Drivers	4
CDHS.s1.3.5.	ETC Drivers	5
CDHS.s1.3.6.	SD Card Drivers (includes load switches) FDIR	4
CDHS.s1.3.7.	Bluetooth drivers – UART +load switches	2
CDHS.s1.3.8.	DRB Interface + load switches	5
CDHS.s2.	Software DGTB	2
CDHS.s2.1	Application layer DGTB	2
CDHS.s2.2	Hardware abstraction layer	2
CDHS.s2.3	Service Layer DRB	2
CDHS.s2.3.1.	Deployment Solar Panels	2
CDHS.s2.3.2.	AntS interface	2
CDHS.s2.3.3.	Debug UART (see above)	2
CDHS.s2.3.4.	Debug LED drivers	9
Total system		
ID	**Description**	**TRL**
CDHS	CDHS	2

Table 2-2: Changelog component table

Changelog date	Component ID	Description

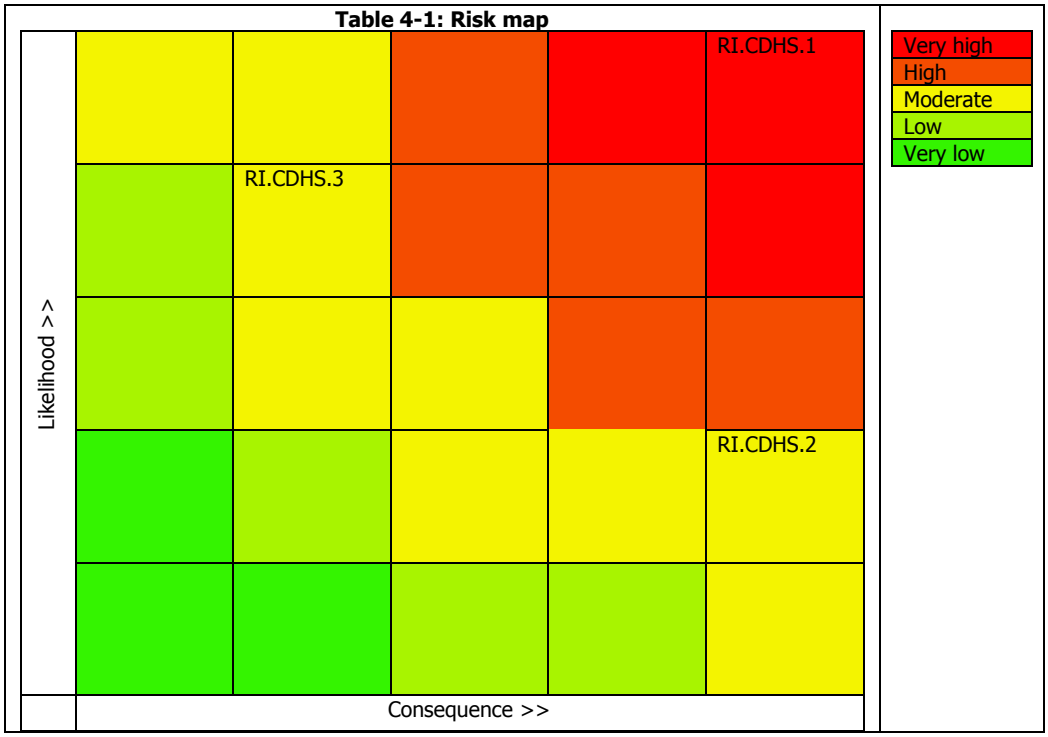
3 Risk identification

Table 3-1: Risk identification

ID				Owner
RI.CDHS.1.	**Description**	Lacking resources in Software development		F Bräuer
	Likelihood	5	Currently Occurring	
	Consequence	5	Without software the S/C cannot fly	
	Status	Ongoing		
RI.CDHS.2.	**Description**	Lacking resources in Hardware development		
	Likelihood	2	Possible lack in the future	
	Consequence	5	Without hardware the S/C cannot fly	
	Status	Ongoing		
RI.CDHS.3.	**Description**	Not meeting performance specifications and requirements		
	Likelihood	4		
	Consequence	2		
	Status	Ongoing		

4 Risk map

Table 4-1: Risk map



5 Mitigation

Table 5-1: Risk mitigation table

ID	Mitigation plan
RI.uPs+.1.	Inform project management
RI.uPs+.2.	Inform project management
RI.uPs+.3.	Revise and redefine conflicting and ambiguous requirements and specifications

6 Conclusion and recommendations

The main identified risk is the development of the on-board software. It is advised to inform project management and increase the resource in the area.

Bibliography

[1] F. Bräuer, "Command and data handling for DelFFi." Literature Study - TU Delft, November 2014.

[2] "Delfi space project website." http://www.delfispace.nl/. Accessed: 04-05-2015.

[3] J. Guo, "Delffi requirements and configuration list 3.0." DFF-TUD-SE-1110[3.0], December 2013.

[4] "QB50 mission objectives." https://www.qb50.eu/index.php/project-description-obj/mission-objectives. Accessed: 24-05-2015.

[5] S. Fasoulas, T. Schmiel, R. Bauman, M. Hörenz, F. U. Hammer, K. Bockstahler, and J. Witt, "New miniaturized and space qualified gas sensors for fast response in situ measurements," in *Proc. 40th Int. Conf. on Environmental Systems AIAA 2010–6147 (Barcelona,)*, pp. 889–904, 2010.

[6] O. Rossman, J. Heisig, and T. Schmiel, *QB50 FIPEX Science Unit ICD*. Technische Universität Dresden, 2.3 ed., August 2014.

[7] E. Gill, P. Sundaramoorthy, J. Bouwmeester, B. Sanders, and C. Science, "Formation flying to enhance the QB50 space network," *Small Satellite Systems and Services Symposium*, p. 3, June 2010.

[8] J. Guo, "Phase-B study on formation flight within qb50." Internal DelFFi documentation, April 2013.

[9] A. Deeb, "Formation flying maintenance for the DelFFi mission." Master Thesis in preparation, September 2015.

[10] I. Krusharev, R. Poyck, B. Zandbergen, A. Cervone, and Q. Bellini, "Cubesat micropropulsion systems for extending the capabilities of academic projects," in *Proc. 65th International Astronautical Congress, Toronto, Canada*, 2014. IAC-14.C4.6.1.x23854.

[11] J. Guo, "Phase-A study on formation flight within qb50." Internal DelFFi documentation, May 2012.

[12] R. Schoemaker, *Robus and Felxible CDHS on board the DelFFI Formation Flying Mission*. Msc thesis, TU Delft, 2014.

[13] Texas Instruments, *MSP430F241X Mixed Signal Microcontroller Data Sheet*.

[14] M. Sallam, "A reliable, fault-tolerant storage system for delffis nano-satellites," Master's thesis, The Hague University of Applied Sciences, June 2014.

[15] A. Franciscus and C. Sander, *MSc THESIS Fault-Tolerant On-Board Computer Software for the Delfi-n3Xt Nanosatellite*. Msc thesis, TU Delft, 2012.

[16] "7-bit, 8-bit, and 10-bit i2c slave addressing." http://www.totalphase.com/support/articles/200349176-7-bit-8-bit-and-10-bit-I2C-Slave-Addressing. Accessed: 25-05-2015.

[17] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[18] L. Wiley, ed., *Applied Space Systems Engineering*. Learning Solutions, 2009.

[19] E. Gill, "Ten theses on the fundamentals of systems engineering research." AE4S01 Space System Engineering Course Literature, January 2013.

[20] DoD, "Mil-std-498 system/subsystem specification sss," tech. rep., Department of Defence, 1994.

[21] D. Buede, *The Engineering Design of Systems: Models and Methods Modes*. Wiley, 2 ed., February 2009.

[22] C. Wasson, *System Analysis, Design, and Development: Concepts, Principles, and Practices*. Wiley, 1 ed., March 2006.

[23] C. Bernal and M. V. Bolhuis, "Releasing the Cloud: A Deployment System Design for the QB50 CubeSat Mission," in *Small Satellite Conference*, pp. 1–6, 2012.

[24] E. Jansen and T. van Wees, "Delffi propulsion system requirements." DFF-TUD-PROP-REQ, April 2015.

[25] A. I. L. Telgie, "Design and analysis of the delffi solar energy supply," Master's thesis, TU Delft, July 2014.

[26] D. L. Oltrogge, A. Graphics, C. Springs, K. Leveque, and M. Park, "An Evaluation of CubeSat Orbital Decay," *25th Annual AIAA/USU Conference on Smallsats*, 2011.

[27] J. Guo, "Delffi mass budget," April 2014. DFF-TUD-BU-1112.

[28] D. Hathaway, "Solar cycle prediction." http://solarscience.msfc.nasa.gov/predict.shtml. Accessed: 07-04-2015.

[29] W. J. Larson and J. R. Wertz, *Space Mission Analysis and Design*. 1999.

[30] T. Flohrer, H. Krag, and H. Klinkrad, "Assessment and categorization of TLE orbit errors for the US SSN catalogue," in *AMOS 2008 Proceedings*, 2008.

[31] "Relected examples of national laws governing space activities: Netherlands." http://www.unoosa.org/oosa/en/SpaceLaw/national/netherlands/space_activities_actE.html. Accessed: 12-04-2015.

[32] R. Reinhard and C. Asma, "Qb50 mission objectives," *5th QB50 Workshop proceedings*, January 2013.

[33] E. Perez, "Soyuz CSG user manual," March 2012.

[34] J. Guo, "Delffi power budget," April 2014.

[35] ISIS, "Ants datasheet," March 2011. DFF-ISS-1144.

[36] A. Denis, C. Asma, C. Bernal, R. Chaudery, Z. de Groot, J. Guo, D. Kataria, D. Masutti, R. Reinhard, T. Scholz, G. Shirville, F. Singaraya, B. Taylor, P. Testani, J. Thoemel, and W. Weggelaar, "QB50 system requirements and recommendations," February 2015.

[37] M. Haghayegh, "Design, implementation, testing of the attidude control algorithm of DelFFi." Master Thesis in preparation, August 2015.

[38] "Rosetta status report: close flyby navigation issues." http://blogs.esa.int/rosetta/2015/04/01/rosetta-status-report-close-flyby-navigation-issues/. Accessed: 12-04-2015.

[39] T. Uhlig, F. Sellmaier, and M. Schmidhuber, eds., *Spacecraft Operations*. Springer, 2015.

[40] J. Eickhoff, ed., *Onboard Computers, Onboard Software and Satellite Operations - An Introduction*. Springer, 2012.

[41] N. Holsti and M. Paakko, "Towards advanced FDIR components," 2001.

[42] A. Guiotto, M. Martelli, and C. Paccagini, "SMART-FDIR: use of Artificial Intelligence in the implementation of a Satellite FDIR," 2003.

[43] D. Evans and M. Merri, "OPS-SAT: An ESA Nanosatellite for Accelerating Innovation in Satellite Control," pp. 1–11, May 2014.

[44] J. Guo, "Risk analysis and mitigation plan," tech. rep., TU Delft, December 2014. CDR Document.

[45] "Space environment." http://www.esa.int/Our_Activities/Space_Engineering_Technology/Space_Environment/Space_environments_and_effects. Accessed: 04-05-2015.

[46] "Total Ionizing Dose (TID) effects." http://radhome.gsfc.nasa.gov/radhome/tid.htm. Accessed: 22-06-2015.

[47] J. Bouwmeester, "Delfi C3 mission review and lessons learnt," tech. rep., TU Delft, 2010.

[48] M. Swartwout, "The first one hundred cubesats: A statistical look," *Jorunal of Small Satellites*, pp. 213–233, 2014.

[49] J. Bouwmeester, M. Langer, and E. Gill, "Results and analysis from a survey on the reliability and performance of cubesat bus interfaces." To be published at: / Provided by J. Bouwmeester on 20.04.2015.

[50] M. Stamatelatos and J. Caraballo, "Fault Tree Handbook with Aerospace Applications," p. 218, 2002.

[51] J. Bouwmeester, L. Rotthier, C. Schuurbiers, W. Wieling, G. van der Horn, F. Stelwagen, E. Timmer, and M. Tijissen, "PRELIMINARY RESULTS OF THE DELFI-N3XT MISSION," in *Proceedings of the Small Satellites and Services Symposium 2014*, pp. 1–15, 2014.

[52] E. D. V. Breukelen, R. J. Hamann, and E. G. Overbosch, "Qualitative fault tree analysis applied as a design tool in a low cost satellite design : Method and lessons learned," no. December 2005, 2006.

[53] "Failure mode and effect analysis." http://www.hq.nasa.gov/office/codeq/software/ComplexElectronics/techniques/fmea.htm. Accessed: 14-05-2015.

[54] K. M. Brumbaugh and E. G. Lightsey, "Application of Risk Management to University CubeSat Missions," *Journal of Small Satellites*, vol. 2, no. 1, pp. 147–160, 2013.

[55] GomSpace, "Nanopower p-series datasheet p31u v8.0," September 2013. DFF-GOM-IC-1133.

[56] S. L. Pfleeger and J. M. Atlee, *Software Engineering Theory and Practice*. Prentice Hall, 2010.

[57] G. Aalbers, "Isis VHF / UHF transceiver," March 2012. DFF-ISS-IC-1137.

[58] P. Rodrigues, "QB50 GAMANET interface control document," Novemver 2013. DFF-TEK-1109.

[59] T. Van Boxtel, "Thermal modelling and design of the delffi satellites," Master's thesis, TU Delft, Delft University of Technology, 2015.

[60] J. Eickhoff, *Simulating Spacecraft Systems*. Springer, 2009.