

TALLINN UNIVERSITY OF TECHNOLOGY
Department of Software Science

Maj-Annika Tammisto, 162768IABM

**A RISK-BASED APPROACH TO STUDENT
SATELLITE SOFTWARE QUALITY
ASSURANCE ON THE EXAMPLE OF THE
TTU 100 SATELLITE MISSION CONTROL
SOFTWARE**

Master`s thesis

Supervisor: Evelin Halling
MSc

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Tarkvarateaduse instituut

Maj-Annika Tammisto, 162768IABM

**RISKIPÕHINE LÄHENEMINE
TUDENGISATELLIIDI TARKVARA
KVALITEEDI TAGAMISELE TTU 100
SATELLIIDI MISSIOONJUHTIMISE
TARKVARA NÄITEL**

Magistritöö

Juhendaja: Evelin Halling
MSc

Tallinn 2018

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Maj-Annika Tammisto

07.05.2018

Abstract

The purpose of this thesis is to create a methodical quality assurance plan for assuring the quality of communication between the Mission Control Software (MCS) and Electrical Power Supply (EPS) of TTU 100 Satellite that is designed, built and launched as part of the TTU Mektory Nanosatellite Programme. Quality assurance is an important topic in student satellite projects because of the high risk of mission failure that is associated with them.

The methodology to be used for achieving the purpose is mainly based on syllabi of the International Software Testing Qualifications Board (ISTQB) and includes risk assessment, test process definition and usage of different testing techniques. In addition to that, best practices from other spacecraft projects are considered and due to that model-based testing is implemented as part of the Common Verification and Validation Environment (CVE) of the MCS.

The limited resources of student projects such as building TTU 100 satellite is taken into account when planning the structure of this thesis, therefore the thesis will not concentrate on how spacecraft software quality is to be assured in perfect conditions. The actual situation, including limited manpower, time, budget and lack of experience is considered when moving towards the purpose of this thesis.

The basis of the quality assurance plan consists of two important building blocks: the requirements that are defined for MCS in different documents and technical risks that are identified, assessed and covered with risk mitigation activities. The quality assurance plan exceeds the scope of traditional software test strategy and test plan documents as there are requirements and risks that cannot be verified or mitigated with functional and non-functional testing only.

The quality assurance plan structure shall be re-usable for other MCS modules and potentially also other software to be developed in the TTU Mektory Nanosatellite Programme or other student satellite projects.

This thesis is written in English and is 40 pages long, including 5 chapters, 9 figures and 14 tables.

Annotatsioon

Riskipõhine lähenemine tudengisatelliidi tarkvara kvaliteedi tagamisele TTÜ 100 satelliidi missioonijuhtimise tarkvara näitel

Käesoleva magistritöö eesmärk on luua TTÜ Mektory Satelliidiprogrammi raames ehitatava TTÜ 100 satelliidi maapealse missioonijuhtimise tarkvara (MCS) ning satelliidi pardal oleva elektroonilise toiteploki (EPS) omavahelise suhtluse kvaliteedi tagamiseks metoodiline kvaliteedi tagamise plaan. Kõrgendatud tähelepanu pööramine kvaliteedi tagamisele on tudengite satelliidiprojekti juures oluline, kuna sarnaste projektidega kaasneb kõrge missiooni ebaõnnestumise risk.

Töö eesmärgi saavutamiseks kasutatav metoodika põhineb peamiselt International Software Testing Qualifications Board (ISTQB) poolt väljastatud juhenditel ja sisaldab riskihindamist, testiprotsessi kirjeldamist ning erinevate testitehnikate kasutamist. Sellele lisaks arvestatakse teiste kosmosetööstuse projektide juures kasutatava praktikaga ning rakendatakse missioonijuhtimise tarkvara testkeskkonna (CVE) loomisel mudelipõhist testimist.

Töös arvestatakse tudengiprojektidele iseloomulike piiratud ressursidega, mille tõttu töös ei keskenduta sellele, kuidas kosmosetööstuses kasutatava tarkvara kvaliteedi tagamine ideaalsetes tingimustes toimuma peab, vaid proovitakse leida optimaalseim lahendus TTÜ 100 satelliidi ehitamise projekti ressursse ja iseärasusi arvesse võttes.

Kvaliteedi tagamise plaan põhineb kahel olulisel sisendil: missioonijuhtimise tarkvarale kehtivad nõuded ning tuvastatud, hinnatud ning vastumeetmetega kaetud tehnilised riskid. Selle tulemusena on kvaliteedi tagamise plaan laiapõhjalisem kui traditsioonilised testistrateegia ja testiplaani dokumendid, kuna see arvestab ka nõuete ning riskidega, mida ei ole võimalik vaid funktsionaalsete ja mittefunktsionaalsete testidega katta.

Käesoleva töö raames loodava kvaliteedi tagamise plaani struktuur on kasutatav ka teiste missioonijuhtimise tarkvara moodulite ning potentsiaalselt ka muu TTÜ Mektory Satelliidiprogrammi või teiste tudengisatelliidi programmide raames loodava tarkvara jaoks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 40 leheküljel, 5 peatükki, 9 joonist, 14 tabelit.

List of abbreviations and terms

MCS	<i>Mission Control Software</i>
TRL	<i>Technology Readiness Level</i>
EPS	<i>Electrical Power Supply</i>
ESA	<i>European Space Agency</i>
ID	<i>Identifier</i>
OBC	<i>On-Board Computer</i>
ISTQB	<i>International Software Testing Qualifications Board</i>
Ku-Band	<i>10 GHz band</i>
IQ	<i>Quadrature modulated signal I & Q components</i>
UHF	<i>Ultra-High Frequency</i>
CVE	<i>Common Verification and Validation Environment</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
DOA	<i>Dead on Arrival</i>
MC/DC	<i>Modified Condition/Decision Coverage</i>
OWASP	<i>Open Web Application Security Project</i>
MBT	<i>Model-Based Testing</i>
SUT	<i>System Under Test</i>
RGB	<i>Red, Green, Blue</i>
NIR	<i>Near Infrared</i>
FPGA	<i>Field-Programmable Gate Array</i>
LED	<i>Light-Emitting Diode</i>
PIC	<i>Programmable Interface Controller</i>
API	<i>Application Programming Interface</i>
GUI	<i>Graphical User Interface</i>
TLE	<i>Two-Line Element set</i>

Table of contents

1 Introduction	13
1.1 Theoretical background	13
1.1.1 TTU Mektory Nanosatellite Programme	13
1.1.2 Nanosatellite	14
1.1.3 TTU 100 Satellite Mission Statement	15
1.1.4 Mission Control Software.....	15
1.2 Problem definition	17
1.3 Objective of the Master’s thesis	19
2 Requirement specifications	20
2.1 TRL 6 assessment guidelines	20
2.2 System requirements for the MCS.....	22
2.3 Requirements for the Common Verification and Validation Environment (CVE)	24
3 Risk analysis	25
3.1 Risk identification.....	25
3.2 Risk assessment	28
3.3 Risk mitigation	34
4 Quality Assurance Plan: low-level communication between MCS and EPS.....	37
4.1 Requirements to be fulfilled	38
4.2 Risk mitigation activities to be considered.....	39
4.3 Test process	41
4.3.1 Planning, monitoring and control	41
4.3.2 Analysis and design.....	42
4.3.3 Implementation and execution	43
4.3.4 Evaluating exit criteria and reporting	47
4.3.5 Test closure activities	48
4.4 Compliance check.....	48

5 Summary.....	51
References	53
Appendix 1 – Thread Identification Protocol	55
Appendix 2 – Likelihood and Impact Definition Protocol	57
Appendix 3 – Testing and Verification Plan for Mission Control Software (MCS).....	59
Appendix 4 – Low-level test cases for MCS communication with EPS	67

List of figures

Figure 1 - The space system [1]	13
Figure 2 - CubeSat specifications [3]	14
Figure 3 - General architecture of MCS [4].	16
Figure 4 - Mission status for Hobbyist CubeSats in 2000-2015 [7].....	18
Figure 5 – Technology Readiness Levels Diagram used by ESA [5]	21
Figure 6 – Three types of software risk [11]	26
Figure 7 – Test Cases Overview [18]	44
Figure 8 - DTRON deployment outline [21]	46
Figure 9 – Sample Uppaal model for communication between the MCS and EPS	46

List of tables

Table 1 – TRL requirements for the MCS [5]	21
Table 2 – System requirements for the MCS [1].....	22
Table 3 - Requirements for the Common Verification and Validation Environment (CVE) [8].....	24
Table 4 – Identified risks	27
Table 5 - Identified risks with categories	29
Table 6 – Likelihood scale definition [12]	31
Table 7 - Impact scale definition [12]	31
Table 8 – Risk Assessment Matrix	32
Table 9 – Risk mitigation activities for product risks with risk levels 1-3.....	35
Table 10 - Requirements applicable for MCS integration with EPS.....	38
Table 11 – Risk mitigation activities applicable for MCS integration with EPS	40
Table 12 – Test Conditions.....	42
Table 13 – Compliance check list.....	49
Table 14 – Test Cases	67

1 Introduction

1.1 Theoretical background

1.1.1 TTU Mektory Nanosatellite Programme

The TTU Mektory Nanosatellite Programme is a pilot project started by TTU Mektory Space Centre in the fall semester of 2014. The goal of this programme is to design a space system which consists of a Space Segment (Satellite) and a Ground Segment as shown in Figure 1 [1].

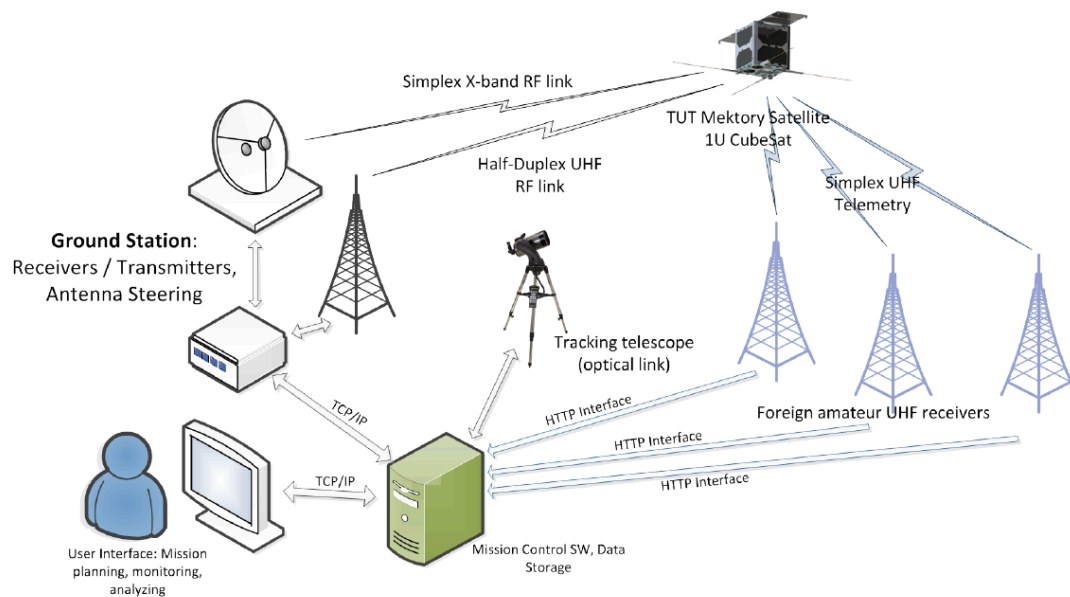


Figure 1 - The space system [1]

There are several teams working on this satellite project, each of which is responsible for a specific field: Electronics, Optics, Software etc. The work of these teams is led by academic supervisors [2].

1.1.2 Nanosatellite

Hundreds of satellites have been launched to space since the historic launch of Sputnik 1 in 1957. The space industry has traditionally produced large and sophisticated aircraft for several decades, but recent advances in technology miniaturization have provided a low-cost and low-power alternative with reduced size and complexity [3].

Created by Stanford and California Polytechnic State Universities in 1999, the CubeSat standard specifies a standard 1U unit as a 10 cm cube ($10 \times 10 \times 10 \text{ cm}^3$) with a mass of up to 1.33 kg [3].

CubeSats are divided to classes based on their mass as shown on Figure 2.

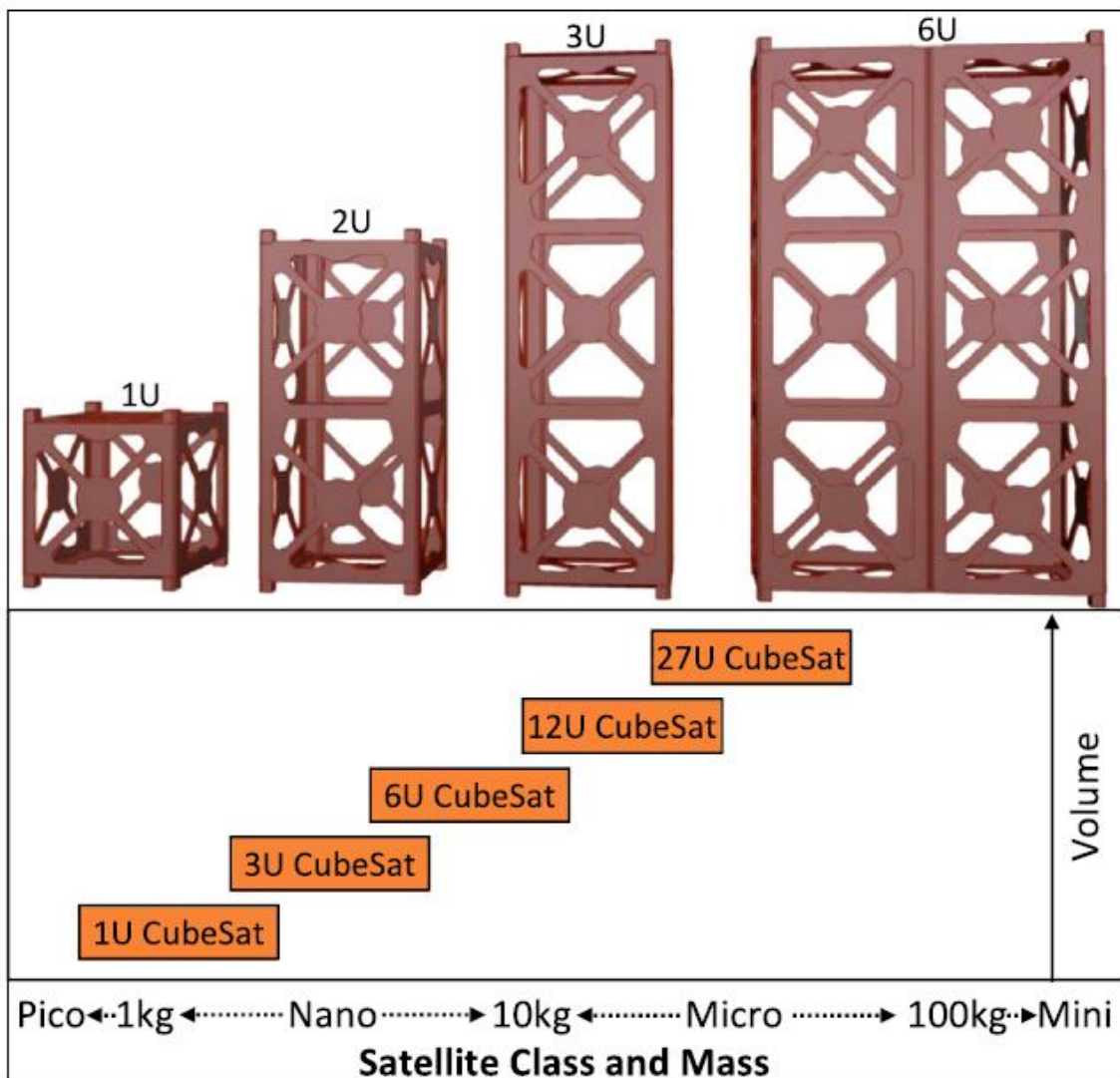


Figure 2 - CubeSat specifications [3]

The official name of the satellite built in the TTU Mektory Nanosatellite Programme is “TTU 100 Satellite” and it is defined as “...a 1U size nanosatellite, according to CalPoly Cubesat Design Specification, on Earth’s Sun Synchronous Orbit (~650km altitude)” [1]. This definition corresponds to the specification on Figure 2, based on which a 1U size CubeSat is indeed a Nanosatellite.

1.1.3 TTU 100 Satellite Mission Statement

The mission statement of TTU 100 Satellite is to monitor the Earth, demonstrating the technology that is necessary for this and conducting different scientific experiments on board the satellite.

The payload of TTU 100 Satellite includes:

- A red, green and blue colour space (RGB) camera that provides coloured images taken with visible light;
- A near-infrared (NIR) camera for monitoring climate and vegetation;
- A field-programmable gate array (FPGA) chip for testing fault tolerance of data communication;
- Light-emitting diodes (LED) and laser diodes for evaluating different methods of sending optical signals.

1.1.4 Mission Control Software

The Mission Control Software of the TTU 100 Satellite (hereinafter referred to as MCS) is a part of the Ground Segment and it consists of five main components:

- Back-office (Spring Boot + AngularJS);
- Public web-based interface (Application Programming Interface (API) + Graphical User Interface (GUI));

- Main calculation and planning module (Mission Planning, Orbit and Contact Prediction, Orbital Data Module, Communication Module, Telemetry Module, Remote Sensing Module, Monitoring);
- Message broker (ActiveMQ);
- Database (Postgres) [4].

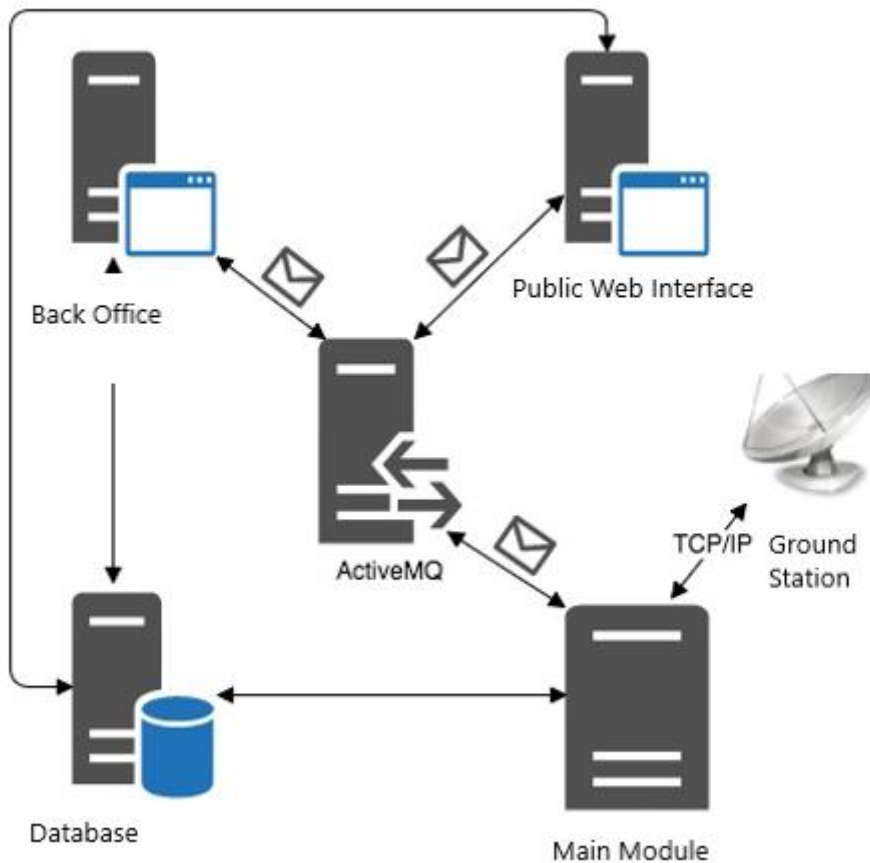


Figure 3 - General architecture of MCS [4].

It enables the following actions:

- building the experiment schedule and showing the results of experiments conducted on board of the satellite;
- exchanging data and files with the satellite;

- sharing information (satellite position over world map, latest time of contact etc.) with the publicity via a public web-based interface;
- receiving ultra-high frequency (UHF) telemetry data from other radio amateurs;
- providing a satellite orbit propagation interface to external clients;
- providing satellite orbit propagation data to the Ground Station system over Transmission Control Protocol/Internet Protocol (TCP/IP) interface;
- providing satellite orbit propagation data to external users for tracking the satellite with optical telescopes [1].

The MCS shall comply with the Technology Readiness Level 6 as specified in the “Technology Readiness Levels Handbook for Space Applications” published by the European Space Agency (ESA) [5].

1.2 Problem definition

The statistics published in Aerospace Conference 2015 [6] states that universities have been showing a constant CubeSat mission failure rate of 40% over a period on 15 years (2000 – 2014) which could indicate that universities are not getting better in achieving mission success.

In 2016 [7] these statistics were specified when the group that was previously referred to as “universities” was divided into four categories of mission developer. New universities working on their first-ever spacecraft project fall into the category of Hobbyists who are showing extremely poor success rates where less than 40% of missions achieve even a subset of their objectives and who are therefore heavily contributing to the overall high mission failure rate of universities.

Mission status for Hobbyist CubeSats in 2000-2015 is illustrated in the graph below, where the mission stages are defined as follows:

0 - (Prelaunch). The mission has been manifested, but has not launched;

1 - (Launched). The mission has launched. Missions lost to launch failure remain at status 1; they are listed as Launch Fail on the charts;

2 - (Ejected). The ejection of the secondary from the launch vehicle has been confirmed. Missions that are ejected, but never contacted, remain at status 2 and are listed as Dead on Arrival (DOA);

3 - (Commissioning). Two-way communication has been established, and the spacecraft is being commissioned for operations. Missions that remain at status 3 are marked as Early Loss;

4 - (Initial operations). The spacecraft has commenced primary operations and are listed as Partial Mission;

5 - (Mission success). Minimum mission success has been achieved; these are marked as Full Mission [7].

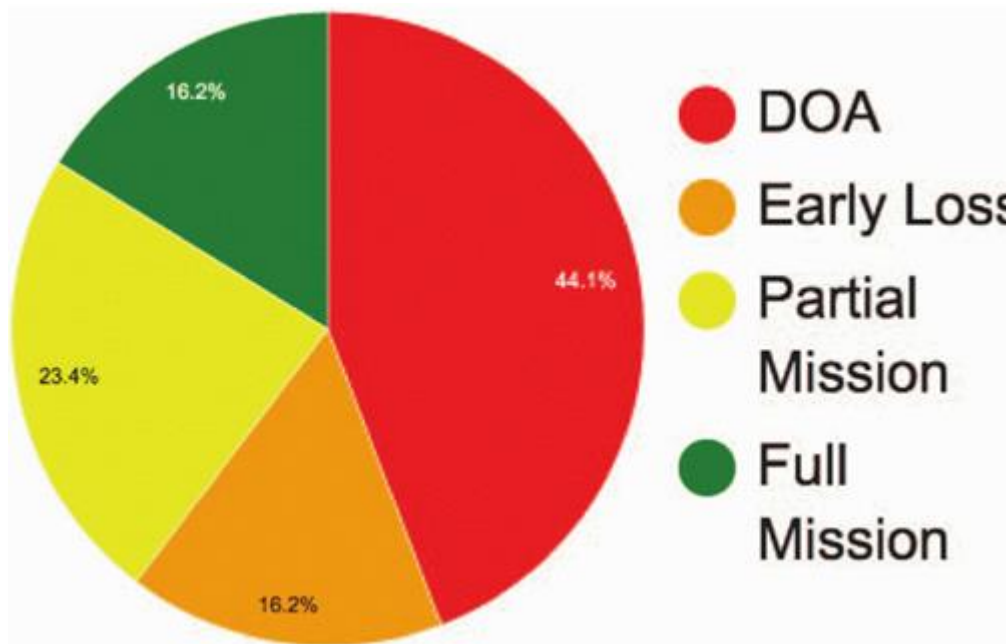


Figure 4 - Mission status for Hobbyist CubeSats in 2000-2015 [7]

A CubeSat Database (<https://sites.google.com/a/slu.edu/swartwout/home/cubesat-database>) has been created and maintained by M. Swartwout, PhD of Saint Louis University and it is claimed to include all CubeSats that have ever flown, showing their mission status and therefore providing up-to-date information about the success and failure rate of CubeSat missions.

By being a pilot project with the goal of launching a first-ever CubeSat developed by TTU, the Mektory Nanosatellite Programme team with no previous experience in spacecraft development falls in the category of Hobbyists. Among other things the Hobbyists are characterized by low-cost, fast-turnaround and a lack of standard practices when it comes to integration and test [7].

The problem addressed in this thesis is the high risk of mission failure of the Mektory Nanosatellite Programme CubeSat TTU 100 based on the statistics of previous first-time university spacecraft missions.

1.3 Objective of the Master's thesis

The overall goal of this thesis is to create a methodical quality assurance plan for assuring the quality of communication between the Mission Control Software (MCS) and Electrical Power Supply (EPS) in order to reduce the risk of mission failure caused by the failure of MCS. The structure of this Plan shall be re-usable for other modules of the MCS.

The thesis shall provide one example of software quality assurance for future student satellite projects, as existing best practices that don't only concentrate on software testing, but on quality assurance in general, are hard to find. Furthermore, if proven to be efficient, the same approach or at least some parts of it could be used in other CubeSat development fields that are not in the scope of this Master's thesis such as Satellite on-board software, Electronics, Optics etc.

The methodology to be used in this thesis is mainly based on syllabi of the International Software Testing Qualifications Board (ISTQB). In addition to that, state of the art from other spacecraft projects is considered with consideration to the fact that the resources that are in disposal of student satellite project teams are considerably smaller than those possessed in spacecraft industry in general.

2 Requirement specifications

Agile methodology is followed in the development process of the MCS, therefore there are no detailed functional requirements that need to be followed. Nevertheless, the MCS needs to comply with the following documents in general:

- The maturity of the MCS shall be consistent with TRL 6 or higher according to the “Technology Readiness Levels Handbook for Space Applications” published by ESA [5].
- The MCS shall comply with the general System Specifications Document for the TTU 100 Satellite [1].
- The MCS shall include a Common Verification and Validation Environment (CVE) as described in the Design Definition File Template Document [8].

The scope of this thesis includes the low-level integration of the MCS with EPS. A general description and working examples of the EPS communication protocol are provided by the EPS development team, but other than this EPS is considered as a black box.

2.1 TRL 6 assessment guidelines

According to the Technology Readiness Levels definition, TRLs are “a set of management metrics that enable the assessment of the maturity of a particular technology and the consistent comparison of maturity between different types of technology - all in the context of a specific system, application and operational environment” [5]. The purpose of using TRLs is to inform management and to support decisions in advanced technology system development projects.

There are 9 levels on the TRL scale used by ESA as illustrated in the figure below.

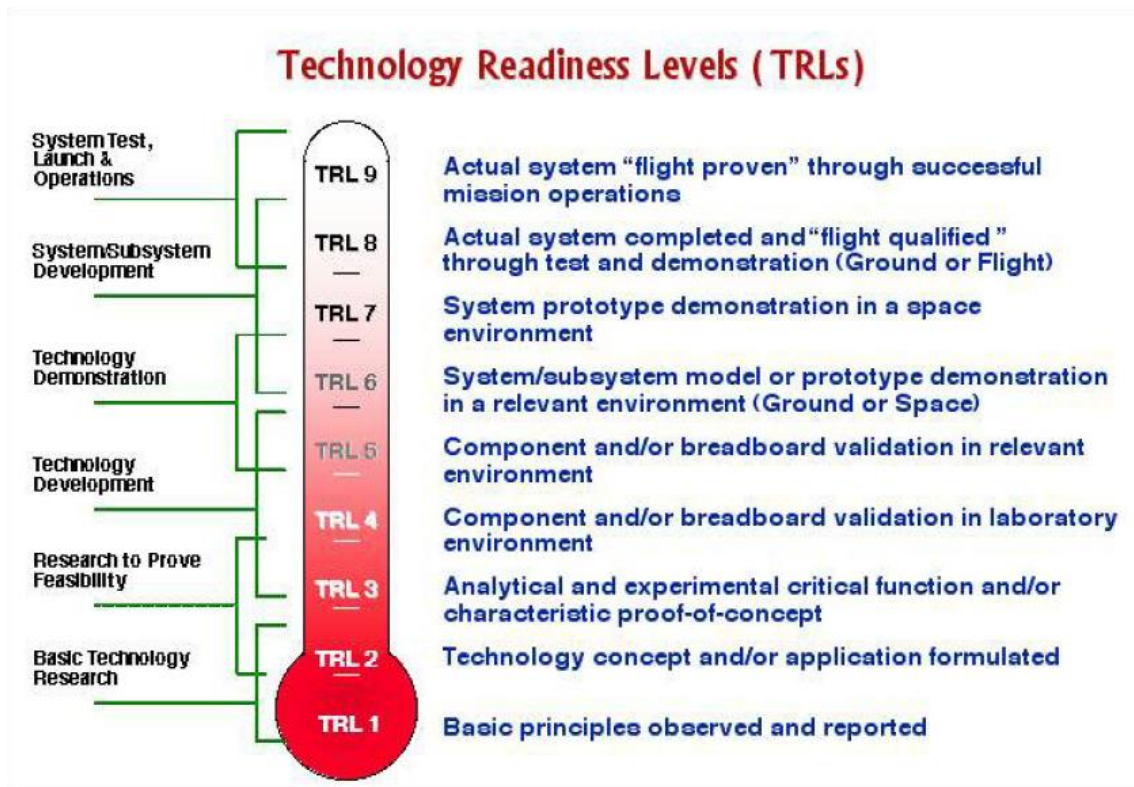


Figure 5 – Technology Readiness Levels Diagram used by ESA [5]

In order to comply with TRL 6, the system has to successfully meet all criteria of all lower TRLs that are applicable for software systems as well as the requirements of TRL 6 [5].

The list of requirements that apply to the MCS are listed in the table below. Requirements that are defined for assessing the readiness of capabilities of new scientific facts or principles are excluded.

Table 1 – TRL requirements for the MCS [5]

ID	Requirement description
TR-1	The new technology, including the design of demonstrations performed and explanation of how the testing environment is relevant to the expected operational environment is clearly described.
TR-2	A document describing in full detail the expected functional and environmental requirements that the new technology must satisfy within the context of the envisaged application is present.

ID	Requirement description
TR-3	Rigorous system-level demonstrations, including testing of key elements individually and/or in integrated fashion, have successfully been performed in a relevant environment and documented.
TR-4	Technical risk (Low, Medium, High) and required effort (Low, Medium, High) to advance to the next TRL level is evaluated.

2.2 System requirements for the MCS

System requirements for the MCS and other sub-systems of the satellite are described in the general system specifications document for the TTU 100 Satellite [1].

Table 2 – System requirements for the MCS [1]

ID	Requirement description
SR-1	The MCS provides user interface for conducting experiments on the satellite and visualizing the downloaded data.
SR-2	The MCS enables the user to build the experiment schedule.
SR-3	The MCS enables the user to exchange data/files with satellite.
SR-4	The MCS chops data into individual packets to be sent to satellite via ground station. Communication with the satellite may happen in burst mode where a number of packets are sent and a number of packets are expected from the satellite.
SR-5	The MCS receives all frames from ground station and extracts data from frames into respective data structures.
SR-6	All 10 GHz band (Ku-band) high speed downlink frames received from satellite shall be stored in ground station data storage in raw Quadrature modulated signal I & Q components (IQ) data format in addition to normal decoded format for a period of 1 month.

ID	Requirement description
SR-7	All of the data downlinked to the ground segment, via UHF and Ku-band downlink channels, shall be stored in a dedicated server for up to 1 year after the completion of the mission.
SR-8	The MCS shall implement communication sessions using priorities so that higher priority communications take precedence over lower level priority communications.
SR-9	The MCS shall enable communication sessions that consist of multiple flybys.
SR-10	The MCS shall assemble the communication frame structure in such a way that time to radio silence (radio eclipse) is considered in satellite expected response scheduling.
SR-11	The MCS shall visualize the satellite position over world map.
SR-12	The MCS shall provide means to send and receive data files form space segment On-Board Computer (OBC).
SR-13	The MCS shall provide interface to assemble high level command sequences into transmit and receive frame sequences to be communicated with satellite at predefined times.
SR-14	The MCS shall provide interface for assembling high level mission descriptions to the satellite.
SR-15	The MCS shall enable describing a terrestrial point to the satellite to track with cameras (regardless of satellite orbital position) and have an option to take a number of pictures with certain delays.
SR-16	The MCS shall enable describing a terrestrial point or sequence of points and satellite orbital positions from where to take pictures with one or both of the satellite cameras.
SR-17	<p>The MCS shall provide a public, web based, user interface with the following functionality:</p> <ul style="list-style-type: none"> • Satellite current position visualization over earth map • Latest time of contact • Latest telemetry status of parameters: TBD • Number of orbits since deployment • Time in orbit since deployment

ID	Requirement description
SR-18	The MCS shall provide an interface to receive UHF telemetry data from other radio amateurs.
SR-19	The MCS shall provide satellite orbit propagation interface to external clients.
SR-20	The MCS shall provide satellite orbit propagation data to ground station system over TCP/IP interface.
SR-21	The MCS shall provide satellite orbit propagation data to external users for tracking the satellite with optical telescopes.

2.3 Requirements for the Common Verification and Validation Environment (CVE)

The overall objective of creating a CVE is to align the verification methodology and approach in various work packages in order to improve the visibility of verification status and to achieve higher productivity through reusing tests at various integration levels [8].

Table 3 - Requirements for the Common Verification and Validation Environment (CVE) [8]

ID	Requirement description
CVE-1	The CVE shall enable stand-alone module level verification.
CVE-2	The CVE shall enable verification of integrated modules.
CVE-3	The CVE shall enable verification of integrated satellite.
CVE-4	The CVE shall provide templates for test case implementation.
CVE-5	The CVE shall provide a common environment for test implementation.
CVE-6	The CVE shall enable re-using of common sequences and functions at various verification stages and integration levels.
CVE-7	The CVE shall enable repeatability of tests.
CVE-8	The CVE shall enable test automation.
CVE-9	The CVE shall enable test plan tracking.

3 Risk analysis

The overall high risk of mission failure described in section 1.2 is addressed by analysing technical risks related to the MCS out of which some, in case of realisation, could potentially lead to overall failure of the Mektory Nanosatellite Programme. Identifying risks that are high and that can, in case of realization, most likely contribute to overall mission failure, will help the team to concentrate their quality assurance resources on appropriate risk mitigation activities.

The method used for this analysis is based on the Risk-Based Testing approach described in the ISTQB Technical Test Analyst Syllabus [9]. ISTQB (International Software Testing Qualifications Board) was founded in November 2002 and is a not-for-profit association based on volunteer work of international testing experts. The main activity of this organisation is the certification of competences in software testing [10].

The risk analysis includes the following three tasks:

- Risk identification;
- Risk assessment;
- Description and planning of risk mitigation activities.

3.1 Risk identification

In the context of this risk analysis, risk is defined as the possibility of an event happening (thread).

There are three types of software risks:

- Project risks: risks that are mainly related to project management, contracts, resources etc.;

- Process risks: risks that are mainly related to planning and development process;
- Product risks: technical risks that are specifically related to the software product (security, performance etc.) [11].

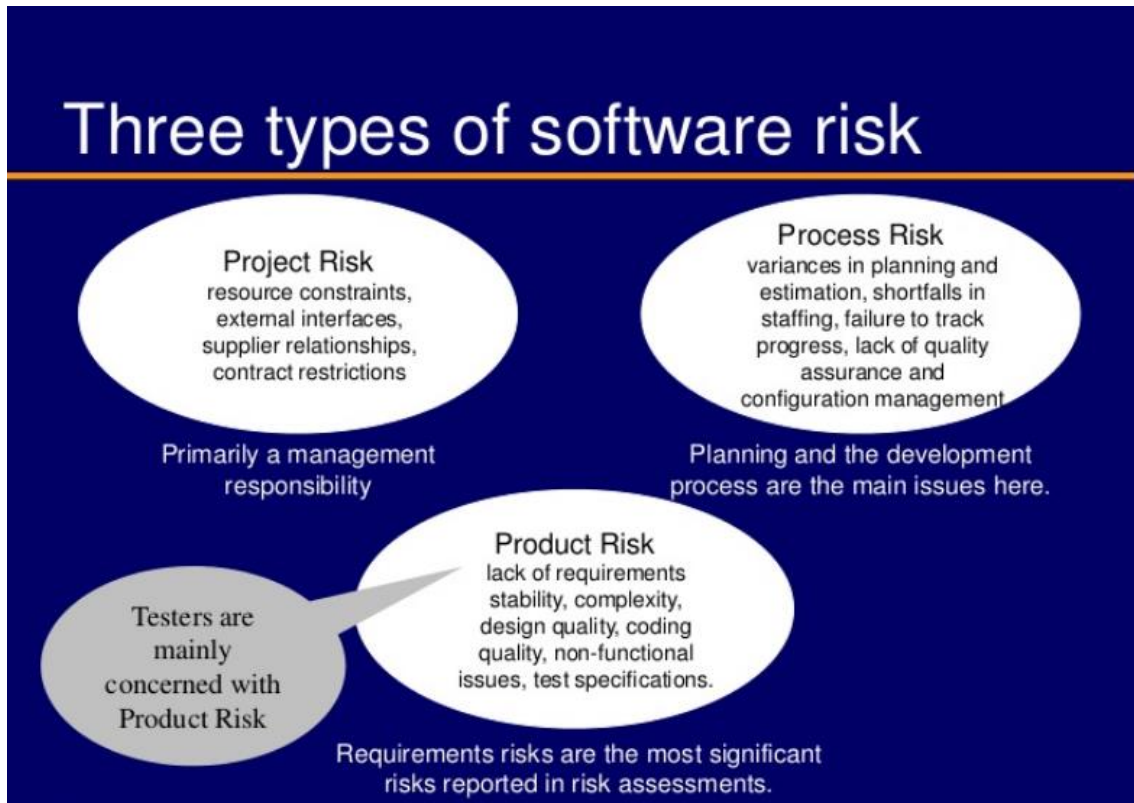


Figure 6 – Three types of software risk [11]

The risk analysis concentrates on product/technical risks in particular, as these are the ones that we can directly influence on the working level. Project and process risks that are identified based on the risk identification process are forwarded to the Academic Supervisor of the MCS team.

All relevant stakeholders, in this case the MCS development team (Academic Supervisor, Developers, Architect and Software Tester) are included in the risk identification process, as advised in the ISTQB Technical Test Analyst Syllabus [9].

The tasks in the risk identification process are:

- Listing all possible threads related to the MCS during a brainstorming event.

Responsibility: the whole MCS team.

- Defining the risk related to every thread listed.

Responsibility: the author of this thesis

- Defining the type of every risk listed.

Responsibility: the author of this thesis

The protocol of the brainstorming event as well as the full list of identified threads is included in Appendix 1 of this thesis. The list of identified risks is presented in the following table.

Table 4 – Identified risks

ID	Thread No.	Risk description
PR-1	1	There may be integration issues between different modules and systems.
PR-2	2	Incorrect or faulty software may be uploaded on the satellite.
PR-3	3	A malicious user may gain access to MCS back-office.
PR-4	4	An error in bit stuffing may cause radio silence.
PR-5	5	The satellite may send us false information.
PR-6	6	A malicious user may gain access to ActiveMQ.
PR-7	7	We may not be able to access the server in case of urgencies.
PR-8	8	We may lose data due to poor reaction of server failure.
PR-9	9	Radio noise caused by amateurs may hinder our communication with the satellite.
PR-10	10	Amateur Telemetry may accidentally be considered as our own by the MCS.
PR-11	11	The MCS may cause physical damage to our antenna.
PR-12	12, 15	Physical damage to the satellite in space may not be identified, communicated or understood through the MCS.

ID	Thread No.	Risk description
PR-13	13	Data may not be stored according to requirements and for the required amount of time.
PR-14	14	Data may accidentally be deleted.
PR-15	16	The Git Repository may accidentally be deleted.
PR-16	17	MCS performance issues may cause delays in sending commands to the satellite.
PR-17	18	The message to/from the satellite or a part of it may become lost when transferring the message from one module or system to the other.
PR-18	19	Docker or the server on which Docker is running may crash unexpectedly.

3.2 Risk assessment

Risk assessment is performed on the basis of the shortlist of MCS-related product risks.

The tasks in the risk assessment process are:

- Categorizing product risks based on the specific area that they influence.

Responsibility: the author of this thesis.

- Defining the likelihood and impact of every risk.

Responsibility: the whole MCS team.

- Creating a risk assessment matrix.

Responsibility: the author of this thesis.

The categories of all identified risks are given in the table below. The following product risk categories are used:

- Reliability risk: the risk related to the software not meeting the specification, requirements or the functionality being inadequate for achieving the expected results;

- Security risk: the risk related to possible unauthorized access, malicious usage or unintentional actions with negative consequences;
- Performance risk: the risk related to software performance.

Risks that are not in one of the above mentioned categories in relation to the MCS are forwarded to the Academic Supervisor and excluded from further analysis.

Table 5 - Identified risks with categories

ID	Category	Risk description
PR-1	Reliability risk	There may be integration issues between different modules and systems.
PR-2	Reliability risk	Incorrect or faulty software may be uploaded on the satellite.
PR-3	Security risk	A malicious user may gain access to MCS back-office.
PR-4	Reliability risk	An error in bit stuffing may cause radio silence.
PR-5	<i>A Product risk related to Satellite on-board software</i>	The satellite may send us false information.
PR-6	Security risk	A malicious user may gain access to ActiveMQ.
PR-7	<i>Planning risk</i>	We may not be able to access the server in case of urgencies.
PR-8	Reliability risk	We may lose data due to poor reaction of server failure.
PR-9	Reliability risk	Radio noise caused by amateurs may hinder our communication with the satellite.
PR-10	Reliability risk	Amateur Telemetry may accidentally be considered as our own by the MCS.
PR-11	Reliability risk	The MCS may cause physical damage to our antenna.
PR-12	Reliability risk	Physical damage to the satellite in space may not be identified, communicated or understood though the MCS.
PR-13	Reliability risk	Data may not be stored according to requirements and for the required amount of time.
PR-14	Security risk	Data may accidentally be deleted.
PR-15	Security risk	The Git Repository may accidentally be deleted.

ID	Category	Risk description
PR-16	Performance risk	MCS performance issues may cause delays in sending commands to the satellite.
PR-17	Performance risk	The message to/from the satellite or a part of it may become lost when transferring the message from one module or system to the other.
PR-18	Reliability risk	Docker or the server on which Docker is running may crash unexpectedly.

Risks are assessed using the qualitative risk assessment method where the likelihood and impact of every risk are measured on a relative scale. The likelihood and impact ratings are transferred to a risk assessment matrix in order to define the risk level of every risk [12]. Risk levels are categorized with a value from 1 to 5, with 1 being the highest risk [9].

By defining likelihood and impact ratings, the following factors are considered:

- Complexity of technology;
- Complexity of code structure;
- Conflict between stakeholders regarding technical requirements;
- Communication problems resulting from the geographical distribution of the development organization;
- Tools and technology;
- Time, resource and management pressure;
- Lack of earlier quality assurance;
- High change rates of technical requirements;
- Large number of defects found relating to technical quality characteristics;
- Technical interface and integration issues [9].

The following likelihood and impact scales are used in this risk assessment:

Table 6 – Likelihood scale definition [12]

Rating	Likelihood	Description
1	Very low	Highly unlikely to occur. May occur in exceptional situations.
2	Low	Most likely will not occur. Infrequent occurrence in other Satellite/Space Technology projects.
3	Moderate	Possible to occur.
4	High	Likely to occur. Has occurred in other Satellite/Space Technology projects.
5	Very high	Highly likely to occur. Has occurred in other Satellite/Space Technology projects and conditions exist for it to occur in this project.

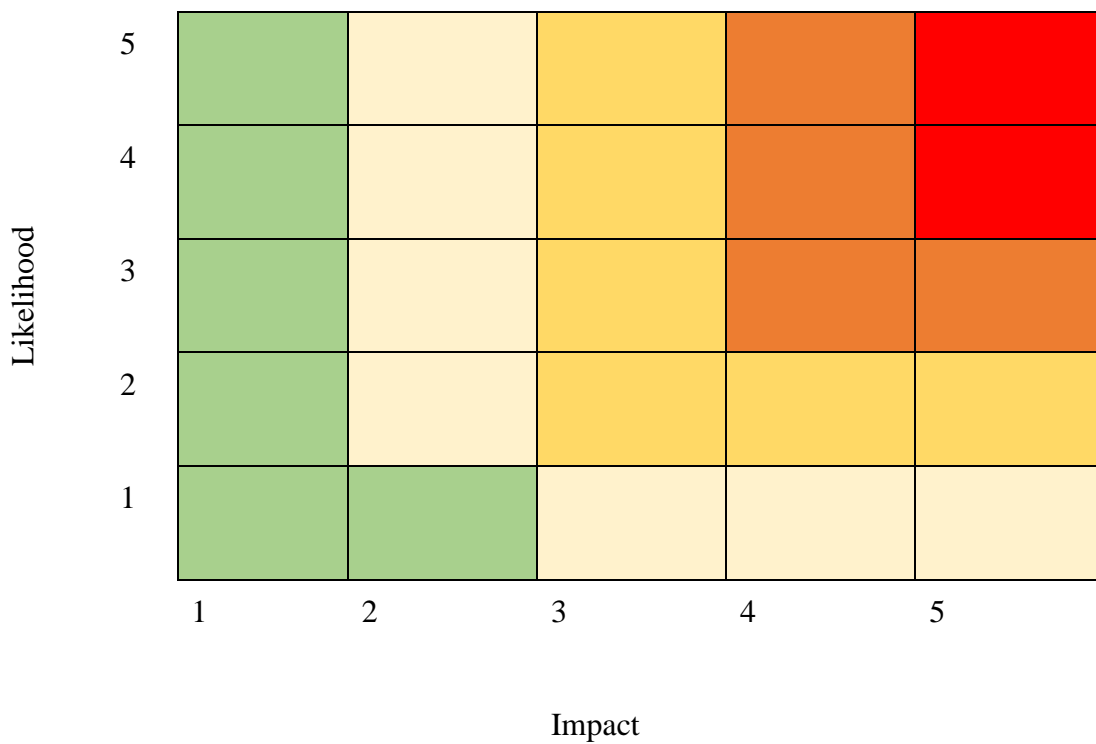
Table 7 - Impact scale definition [12]

Rating	Impact	Description
1	Very low	No increase in budget or schedule. No effect on critical functionality of the system or reputation of the project.
2	Low	May cause a small increase of budget (< 5%) and slight delays in schedule (< 1 week). May slightly affect critical functionality of the system and the reputation of the project.
3	Moderate	May cause a 5-10% increase of budget and a 1-2 week delay in schedule. Probably affects critical functionality of the system and may affect the reputation of the project.
4	High	May cause 10-20 % increase of budget and at least a 2-4 week delay in schedule. Affects critical functionality of the system and probably also the reputation of the project.






Rating	Impact	Description
5	Very high	May cause > 20 % increase of budget and no less than a 4 week delay in schedule. Significantly affects critical functionality of the system and the reputation of the project.

The Risk Assessment Matrix, including risk levels from 1-5 is shown in the table below:

Table 8 – Risk Assessment Matrix



Legend

	Risk Level 1
	Risk Level 2
	Risk Level 3
	Risk Level 4
	Risk Level 5

A full likelihood and impact definition protocol with likelihood and impact associated with every product risk is included in Appendix 2 of this thesis.

Based on their likelihood and impact, product risks are prioritized according to corresponding risk levels as follows:

Risk Level	ID	Category	Risk description
1	PR-2	Reliability risk	Incorrect or faulty software may be uploaded on the satellite.
2	PR-1	Reliability risk	There may be integration issues between different modules and systems.
2	PR-11	Reliability risk	The MCS may cause physical damage to our antenna.
2	PR-12	Reliability risk	Physical damage to the satellite in space may not be identified, communicated or understood though the MCS.
2	PR-17	Performance risk	The message to/from the satellite or a part of it may become lost when transferring the message from one module or system to the other.
3	PR-3	Security risk	A malicious user may gain access to MCS back- office.
3	PR-4	Reliability risk	An error in bit stuffing may cause radio silence.
3	PR-6	Security risk	A malicious user may gain access to ActiveMQ.
3	PR-8	Reliability risk	We may lose data due to poor reaction of server failure.
3	PR-14	Security risk	Data may accidentally be deleted.
3	PR-16	Performance risk	MCS performance issues may cause delays in sending commands to the satellite.

3.3 Risk mitigation

Risk mitigation activity is an activity that is aimed for reducing the likelihood and/or impact of a certain risk. Risk mitigation does not necessarily mean that the realization of all identified and assessed risks need to be avoided.

Considering that the MCS team of the TTU Mektory Nanosatellite Programme consists mostly of students and volunteers who are working on the project besides their studies and day-jobs, it is also not feasible to have every single risk covered with risk mitigation activities. Instead, risks with the highest risk level will be addressed and risks with the lowest risk level will be tolerated, in order to achieve software quality that is sufficient [13] for saying that the critical functionality of the MCS will very likely work as expected.

The risk-based approach allows the team to concentrate on the factors that can with the highest probability lead to the project failure instead of putting their time and effort on less critical things. On the other hand, it will make all identified risks visible for all stakeholders. Therefore it will also be visible which risks are knowingly taken [11].

In the current risk analysis risk mitigation activities are defined for product risks with risk levels 1-3. Risks are regularly evaluated based on additional information gathered as the project unfolds and should a previously tolerated risk reach a risk level where mitigation actions are necessary, the gathered information will be used to implement the actions aimed at decreasing the likelihood or impact of such risk [9].

Table 9 – Risk mitigation activities for product risks with risk levels 1-3

Risk Level	ID	Risk mitigation activities
1	PR-2	<ul style="list-style-type: none"> • Software shall be uploaded to the satellite only after successful flat-sat level tests on the CVE. • Software code to be uploaded to the satellite that is developed by the MCS team is to be tested to 100% Modified Condition/Decision Coverage (MC/DC) as specified in the ISTQB Technical Test Analyst Syllabus [10]. • Software code to be uploaded to the satellite that is not developed by the MCS team is uploaded only after receiving a written confirmation from the developer that the provided version of code is tested and safe to upload. The developer is encouraged to test the code to 100% Modified Condition/Decision Coverage (MC/DC) as specified in the ISTQB Technical Test Analyst Syllabus [10].
2	PR-1	<ul style="list-style-type: none"> • Integration testing shall be performed on the CVE, in which the integration of some components as well as the integration of the whole satellite and the Ground Segment is tested on flat-sat level.
2	PR-11	<ul style="list-style-type: none"> • Specification of the antenna is requested in order to define its physical limits. • Test cases are designed using black-box testing techniques (boundary values, equivalence classes) as specified in the ISTQB Foundation Level Syllabus [18] specifically for identifying potential defects in controlling the antenna.
2	PR-12	<ul style="list-style-type: none"> • An Analysis shall be performed in order to define the possibility of developing a warning system or a set of parameters that could indicate that the satellite has been physically damaged in space. • Test cases are designed using black-box testing techniques (boundary values, equivalence classes) as specified in the ISTQB Foundation Level Syllabus [18] for the warning system, should one be developed.

Risk Level	ID	Risk mitigation activities
2	PR-17	<ul style="list-style-type: none"> • An Architectural Review as specified in the ISTQB Technical Test Analyst Syllabus [10] shall be conducted in order to identify possible places and/or situations where the message or a part of it can potentially be lost. • Test cases shall be designed and tests performed based on the results of this review, using an appropriate testing technique (black-box testing, integration testing etc.) as specified in the ISTQB Foundation Level Syllabus [18].
3	PR-3	<ul style="list-style-type: none"> • User authentication process shall be analysed, defined and implemented. • Security tests shall be designed based on the Open Web Application Security Project (OWASP) Authentication Cheat Sheet [14] and executed.
3	PR-4	<ul style="list-style-type: none"> • Test cases are designed using black-box testing techniques (boundary values, equivalence classes) as specified in the ISTQB Foundation Level Syllabus [14] specifically for identifying potential defects in bit stuffing.
3	PR-6	<ul style="list-style-type: none"> • User authentication process shall be analysed, defined and implemented. • Security tests shall be designed based on the OWASP Authentication Cheat Sheet [14] and executed.
3	PR-8	<ul style="list-style-type: none"> • Server health monitoring possibilities shall be analysed, defined and implemented. • A process of regular server back-ups shall be analysed, defined and implemented.
3	PR-14	<ul style="list-style-type: none"> • Giving write permission to live database is carefully considered. For most team members, read permission shall be enough. • A process of regular database back-ups shall be analysed, defined and implemented.

Risk Level	ID	Risk mitigation activities
3	PR-16	<ul style="list-style-type: none"> • An Architectural Review as specified in the ISTQB Technical Test Analyst Syllabus [9] shall be conducted in order to identify any potential performance issues. • Performance tests shall be designed and performed on the CVE.

4 Quality Assurance Plan: low-level communication between MCS and EPS

EPS (Electrical Power Supply) is a sub-system on the Satellite bus that consists of rechargeable batteries and solar cells and provides electrical supply for all systems and instruments on board. It logs data such as solar cell voltages and currents for all solar cell groups, battery voltage and currents for all battery groups, voltage and current for all power rails, battery temperatures, EPS processor reset event counts, power off-on reset counts, battery charging levels etc. [1].

All data is organised in form of 16-bit registers (2 bytes). There are 101 data registers in total with numbering from 0 to 100. Some of these data registers are editable, others are write-protected (read-only). In addition to that, there are additional special registers for outputs control, transmitting user messages via Lasers, Firmware update and Log reading [14].

EPS receives commands in case they are sent directly to its address 0x04 and replies to these commands [14]. It does not support multi-threading [15].

The following commands are supported:

- Read holding registers (0x03)
- Write holding register (0x06)

- Write multiple registers (0x10)

Reset command is executed when transmitted to circular address 0x55 [14].

4.1 Requirements to be fulfilled

Tables with requirements that the MCS needs to meet are given in sections 2.1, 2.2 and 2.3. The following of these requirements are relevant to the integration of MCS with EPS:

Table 10 - Requirements applicable for MCS integration with EPS

ID	Requirement description
TR-1	The new technology, including the design of demonstrations performed and explanation of how the testing environment is relevant to the expected operational environment is clearly described.
TR-2	A document describing in full detail the expected functional and environmental requirements that the new technology must satisfy within the context of the envisaged application is present.
TR-3	Rigorous system-level demonstrations, including testing of key elements individually and/or in integrated fashion, have successfully been performed in a relevant environment and documented.
TR-4	Technical risk (Low, Medium, High) and required effort (Low, Medium, High) to advance to the next TRL level is evaluated.
SR-3	The MCS enables the user to exchange data/files with satellite.
SR-4	The MCS chops data into individual packets to be sent to satellite via ground station. Communication with the satellite may happen in burst mode where a number of packets are sent and a number of packets are expected from the satellite.
SR-5	The MCS receives all frames from ground station and extracts data from frames into respective data structures.
SR-6	All Ku-band high speed downlink frames received from satellite shall be stored in ground station data storage in raw IQ data format in addition to normal decoded format for a period of 1 month.

ID	Requirement description
SR-7	All of the data downlinked to the ground segment, via UHF and Ku-band downlink channels, shall be stored in a dedicated server for up to 1 year after the completion of the mission.
SR-8	The MCS shall implement communication sessions using priorities such that higher priority communications take precedence over lower level priority communications.
SR-9	The MCS shall enable communication sessions that consist of multiple flybys.
SR-10	The MCS shall assemble the communication frame structure in such a way that time to radio silence (radio eclipse) is considered in satellite expected response scheduling.
CVE-2	The CVE shall enable verification of integrated modules.
CVE-4	The CVE shall provide templates for test case implementation.
CVE-5	The CVE shall provide a common environment for test implementation.
CVE-6	The CVE shall enable re-using of common sequences and functions at various verification stages and integration levels.
CVE-7	The CVE shall enable repeatability of tests.
CVE-8	The CVE shall enable test automation.
CVE-9	The CVE shall enable test plan tracking.

4.2 Risk mitigation activities to be considered

A list of risk mitigation activities needed to consider throughout the whole project is given in section 3.2. The following product risks that require risk mitigation activities are relevant to communication between MCS and EPS:

Table 11 – Risk mitigation activities applicable for MCS integration with EPS

Risk Level	ID	Risk mitigation activities
2	PR-1	<ul style="list-style-type: none"> Integration testing shall be performed on the CVE, in which the integration of some components as well as the integration of the whole satellite and the Ground Segment is tested on flat-sat level.
2	PR-12	<ul style="list-style-type: none"> An Analysis shall be performed in order to define the possibility of developing a warning system or a set of parameters that could indicate that the satellite has been physically damaged in space. Test cases are designed using black-box testing techniques (boundary values, equivalence classes) as specified in the ISTQB Foundation Level Syllabus [16] for the warning system, should one be developed.
2	PR-17	<ul style="list-style-type: none"> An Architectural Review as specified in the ISTQB Technical Test Analyst Syllabus [9] shall be conducted in order to identify possible places and/or situations where the message or a part of it can potentially be lost. Test cases shall be designed and tests performed based on the results of this review, using an appropriate testing technique (black-box testing, integration testing etc.) as specified in the ISTQB Foundation Level Syllabus [16].
3	PR-4	<ul style="list-style-type: none"> Test cases are designed using black-box testing techniques (boundary values, equivalence classes) as specified in the ISTQB Foundation Level Syllabus [16] specifically for identifying potential defects in bit stuffing.
3	PR-16	<ul style="list-style-type: none"> An Architectural Review as specified in the ISTQB Technical Test Analyst Syllabus [9] shall be conducted in order to identify any potential performance issues. Performance tests shall be designed and performed on the CVE.

4.3 Test process

Testing of low-level communication between MCS and EPS follows the fundamental test process as described in the ISTQB Test Analyst syllabus [17]. The activities are:

- Planning, monitoring and control;
- Analysis and design;
- Implementation and execution;
- Evaluating exit criteria and reporting;
- Test closure activities.

4.3.1 Planning, monitoring and control

The general MCS Test Plan included in Appendix 3 is followed therefore no separate Test Plan is necessary for testing MCS integration with EPS.

The following metrics are monitored throughout the whole test process:

- The number of identified product risks covered with tests;
- The number of identified product risks not covered with tests;
- The total number of test cases defined;
- The number of test cases automated;
- The number of test cases executed;
- The number of defects reported.

Accurate metrics indicate whether change (for example additional test effort in specific areas) is needed [17].

4.3.2 Analysis and design

Test analysis defines test conditions, more specifically what is to be tested. Test conditions are viewed as detailed measures and targets for success that are identified by analysis of the test basis, test objectives and product risks [17].

The following general test conditions are specified for testing the low-level communication between MCS and EPS:

Table 12 – Test Conditions

ID	Requirement	Product Risk	Test Condition
CO-1	TR-3 SR-3 SR-4 SR-5 SR-8 SR-9 SR-10 CVE-2 CVE-4 CVE-5 CVE-6 CVE-7 CVE-8 CVE-9	PR-1 PR-4 PR-12 PR-16 PR-17	Send a valid command that is supported by EPS and validate the response.
CO-2	TR-3 SR-3 SR-4 SR-5 SR-8 SR-9 SR-10 CVE-2 CVE-4 CVE-5 CVE-6 CVE-7 CVE-8 CVE-9	PR-1 PR-4 PR-12 PR-16 PR-17	Send an invalid command that is supported by EPS and validate the error.

ID	Requirement	Product Risk	Test Condition
CO-3	TR-3 SR-3 SR-4 SR-5 SR-8 SR-9 SR-10 CVE-2 CVE-4 CVE-5 CVE-6 CVE-7 CVE-8 CVE-9	PR-1 PR-4 PR-12 PR-16 PR-17	Send a valid command that is not supported by EPS and validate the error.
CO-4	TR-3 SR-3 SR-4 SR-5 SR-8 SR-9 SR-10 CVE-2 CVE-4 CVE-5 CVE-6 CVE-7 CVE-8 CVE-9	PR-1 PR-4 PR-12 PR-16 PR-17	Send >1 concurrent commands and validate the response to the first command sent.
CO-5	SR-6 SR-7	(PR-13)	Perform storage testing and verify that data is stored according to requirements.

4.3.3 Implementation and execution

Test implementation involves a detailed description of test environment (in this thesis referred to as the CVE), test data and test cases.

In order to achieve high test coverage, satellite software design process evolves towards model-based approaches. Considering verification and validation, there are two classes of approach:

- Based on human operation and expertise and may be qualified rather empirical even though they are computer aided.
- Automated and may be based on formal notation tools and methods to produce, or to check the properties of the embedded software [18].

The figure below illustrates the difference between the number of test cases manually generated by testers and the number of test cases generated by using a Model-Based Testing (MBT) tool called SpecExplorer on the Earth Sensor Module of Galileo IOV.

EarthSensor	FMEA/MBT
Test Cases (generated by testers)	13
Test Model Description	SpecExplorer
Test Model States	49
Test Model Transitions	82
Test Cases generated	46
Test Cases States	219
Test Cases Transitions	173

Figure 7 – Test Cases Overview [18]

As visible from the numbers in the figure, a MBT tool is highly more efficient in generating test cases and therefore, also assuring a better test coverage.

Nevertheless, using model-based approach in designing a CVE for the Mektory Nanosatellite Program, the following aspects need to be considered:

- There is a lack of open-source MBT tools on the market that would be suitable for testing spacecraft software in terms of allowing behaviour modelling of the system, low-level and high-level testing and being able to communicate with System under Test (SUT) either directly or via an Adapter.

- The usefulness of MBT generated tests is strongly related to the quality of the model that has been defined by the tester. In case the model does not fully reflect the behaviour of the SUT, test coverage will actually be lower than presumed.
- The number of test cases generated by the MBT might explode quite fast. Considering the limited resources of a student project, maintaining, running and evaluating these tests might require more man-hours than the team can afford to spend for this purpose.

The practice of using model-based testing in the verification and validation process of spacecraft software is considered when designing the CVE, but instead of using a dedicated MBT tool, an integrated tool environment for modelling, validation and verification of real-time systems called Uppaal together with the DTRON framework is used. In Uppaal, real-time systems are modelled as networks of timed automata and extended with different data types (bounded integers, arrays, etc.) [19].

Uppaal contains a model checking engine that allows the user to run the model and to identify any deadlocks that it contains in order to avoid modelling errors. The tool can be used free of charge for academic purposes [19].

DTRON is a framework for model-based testing that extends Uppaal and the online test execution tool TRON, enabling coordination, synchronization and online distributed testing [20]. When DTRON executes the xml model created with Uppaal, it will intercept the prefixed Uppaal channel synchronizations within the model. Spread toolkit/server is used for forwarding intercepted synchronizations to the SUT [21].

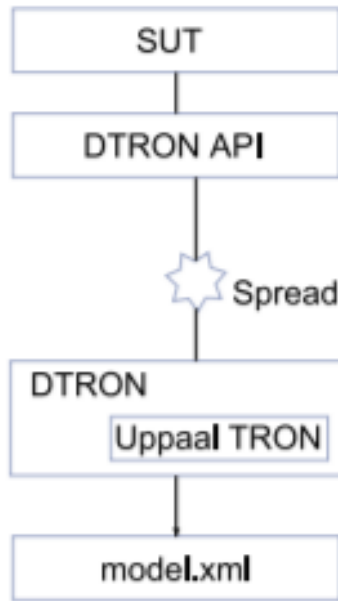


Figure 8 - DTRON deployment outline [21]

Pre-defined test cases are used for modelling different positive and negative scenarios and their expected outcomes. Low-level test cases for MCS communication with EPS are listed in Appendix 4. Whenever the expected outcome is not achieved, it is considered to be a potential defect.

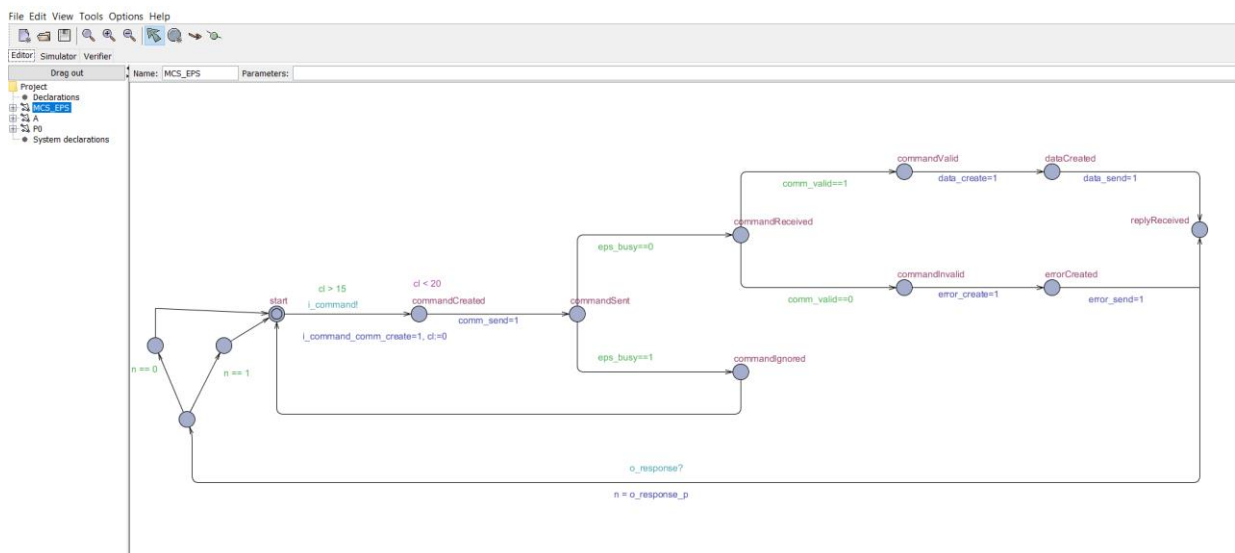


Figure 9 – Sample Uppaal model for communication between the MCS and EPS

Model-based testing with Uppaal and DTRON form an important part of the CVE. No responses shall be mocked and only real software shall be used as SUT in order to keep the testing environment as similar to the expected operational environment as possible.

Once the test object (SUT) is delivered and the entry criteria to test execution is satisfied, tests will be executed. The following entry criteria applies to test execution for the communication between the MCS and EPS:

- Test cases are designed.
- The SUT is modelled in Uppaal.
- The CVE (Uppaal model, DTRON, SUT) is ready for usage.
- The defect tracking tool is in place [17].

4.3.4 Evaluating exit criteria and reporting

Defect reporting is an ongoing process in which defects are reported as soon as possible.

Defects are prioritized and high priority defects are addressed prior to those with lower priority. High priority defects can be:

- Those that can potentially cause the realization of a product risk with risk level 1-3.
- Those that can potentially change the risk level 4 or 5 of a product risk to 1-3.
- Those that are not considered in the risk assessment, but can significantly affect critical functionality of the system and the reputation of the project.

The testing of MCS communication with EPS is considered to be complete if the following exit criteria is met:

- All defined test cases are executed;
- All identified high priority defects are fixed;
- All identified low priority defects are reported;

- A process for regression testing is in place.

4.3.5 Test closure activities

The following test closure activities shall be considered as suggested in the ISTQB Test Manager Syllabus:

- Test completion check - ensuring that all test work is indeed concluded: all planned tests are either run or deliberately skipped and all known defects are either fixed and verified, deferred for a future release, or accepted as permanent restrictions;
- Handover of test artefacts - delivering valuable work products to those who need them. For example, known low priority defects deferred or accepted are communicated to those who will use and support the use of the system;
- Lessons learned - performing a retrospective meeting where good practices and things that need improving can be documented;
- Archiving results, logs, reports, other documents and work products in a dedicated system.

4.4 Compliance check

Requirements to be met and risk mitigation activities to be considered were listed in sections 4.1 and 4.2 of this thesis. This section will verify the compliance of the quality assurance plan with the defined requirements and risk mitigation activities.

Table 13 – Compliance check list

ID	Considered in the quality assurance plan?	Comment
TR-1	Yes	Model-based testing with Uppaal and DTRON form an important part of the CVE. No responses shall be mocked and only real software shall be used as SUT in order to keep the testing environment as similar to the expected operational environment is as possible.
TR-2	Yes	Requirements are presented in section 4.1.
TR-3	Yes	Test conditions are presented in section 4.3.2.
TR-4	Yes	Risks together with their risk levels and risk mitigation activities are presented in section 4.2.
SR-3	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
SR-4	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
SR-5	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
SR-6	Yes	Covered with Test Condition CO-5.
SR-7	Yes	Covered with Test Condition CO-5.
SR-8	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
SR-9	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
SR-10	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-2	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-4	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-5	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-6	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-7	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
CVE-8	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.

ID	Considered in the quality assurance plan?	Comment
CVE-9	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
PR-1	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
PR-12	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
PR-16	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
PR-17	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.
PR-4	Yes	Covered with Test Conditions CO-1, CO-2, CO-3 and CO-4.

5 Summary

The purpose of this thesis was to create a methodical quality assurance plan for assuring the quality of communication between the Mission Control Software (MCS) and Electrical Power Supply (EPS) of TTU 100 Satellite.

The methodology that was used for achieving the purpose was mainly based on syllabi of the International Software Testing Qualifications Board (ISTQB) and included risk assessment, test process definition and usage of different testing techniques. In addition to that, best practices from other spacecraft projects were taken into account with consideration of available resources in the TTU Mektory Nanosatellite Programme.

The biggest lesson learned while working on this thesis was the absolute necessity of identifying and assessing technical risks and defining risk mitigation activities with the whole team. As in most developer teams, every team member in the MCS team is also working on a specific task in a specific area, making it difficult for them to comprehend the whole project. Great synergy can be achieved and very valuable thread suggestions received during collective brainstorming.

Considering the number of student satellites launched, the author of this thesis was surprised to learn that there is much less best practice information available for satellite software quality assurance than expected. Most of the information that is available concentrates on traditional software testing or even on one software test technique in particular. Based on the risks identified in this thesis the author is convinced that only testing is most likely not enough and additional measures such as reviews, additional analysis etc. are needed.

The author of this thesis would have liked to see the CVE in action before the submission of this thesis. Setting up the CVE, including Uppaal, DTRON, SUT Adapter and SUT is in scope of another Master's thesis and is currently still in process of development. In addition to that, the effectiveness of the risk-based quality assurance approach suggested in this thesis can be truly evaluated once the satellite is on the orbit and we can hopefully confirm that the mission was successful and the software is working as expected.

The next activities include implementing Uppaal and DTRON as a part of the CVE and running the model with pre-defined test cases included in it. Additional quality plans shall

be made based on the structure presented in this thesis, the next one will most likely concentrate on assuring the quality of MCS communication with the satellite on-board communication protocol.

Finally, the author of this thesis would like to thank her supervisor Ms. Evelin Halling for all her support and guidance with this thesis, the MCS team for actively participating in the risk assessment process, Mr. Rauno Gordon, Head of the TTU Mektory Nanosatellite Programme, for his kind and operative assistance with all questions related to the satellite mission in general and last but not least, her extended family for supporting her during the two years of Master's studies and for helping her with assuring the quality of English in this thesis.

References

- [1] R. Adelbert, *TUT-MEKTORY NANOSATELLITE System Requirements Specification TMSS-SYS-RS-01, ver 0.5*, Tallinn, 2016.
- [2] “TTÜ Mektory Satelliidiprogramm,” [Online]. Available: <https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4>. [Accessed 15 August 2017].
- [3] A. Poghosyan and A. Golkar, “CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions,” *Progress in Aerospace Sciences*, vol. 88, pp. 59-83, 2017.
- [4] S. Romanov, *Kuupsatelliidi missioonijuhtimistarkvara arhitektuur, TTU Master Thesis*, Tallinn, 2017.
- [5] TEC-SHS/5551/MG/ap, *Technology Readiness Levels Handbook for Space Applications, revision 6*, ESA, 2009.
- [6] M. Swartwout, “Secondary spacecraft in 2015: Analyzing success and failure,” in *Aerospace Conference, 2015 IEE, 07-14.03.2015*, Big Sky, 2015.
- [7] M. Swartwout, “Secondary spacecraft in 2016: Why some succeed (And too many do not),” in *Aerospace Conference, 2016 IEE, 05 – 12.03.2016*, Big Sky, 2016.
- [8] R. Adelbert, *TUT-Mektory nanosatellite design definition file template TMSS-SYS-MM-02, ver 0.1*, Tallinn, 2016.
- [9] G. Bath, P. Jorgensen and J. Mitchell, *Certified Tester Advanced Level Syllabus Technical Test Analyst, version 2012*, International Software Testing Qualifications Board, 2012.
- [10] “International Software Testing Qualifications Board,” [Online]. Available: <https://www.istqb.org/about-as.html>. [Accessed 11 April 2018].
- [11] P. Gerrard and N. Thompson, “Risk-Based Testing, version 1.0a,” in *EuroSTAR Conference*, Edinburgh, 2002.
- [12] “Qualitative Risk Analysis and Assessment,” [Online]. Available: <https://www.project-management-skills.com/qualitative-risk-analysis.html>. [Accessed 12 March 2018].
- [13] H. Hartmann, F. van der Linden and J. Bosch, “Risk Based Testing for Software Product Line Engineering,” in *Proceedings of the 18th International Software Product Line Conference - Volume 1*, Florence, 2014.
- [14] “OWASP Authentication Cheat Sheet,” [Online]. Available: https://www.owasp.org/index.php/Authentication_Cheat_Sheet. [Accessed 29 April 2018].
- [15] V. Sinivee, *EPS communication protocol and register map*, Tallinn, 2016.
- [16] R. Adelbert, *TUT-Mektory Nanosatellite Satellite TCTM Protocol Description TMSS-SYS-TN-02*, Tallinn, 2016.

- [17] T. Müller, D. FriedenberG and t. I. W. F. Level, *Certified Tester Foundation Level Syllabus, version 2011*, International Software Testing Qualifications Board, 2011.
- [18] J. McKay, M. Smith and E. Van Veenendaal, *Certified Tester Advanced Level Syllabus Test Analyst, version 2012*, International Software Testing Qualifications Board, 2012.
- [19] M. Zeuner, H. Gogl, H.-J. Herpel, G. Willich and M.-F. Wendland, “Testing Satellite on-board Software - A Model Based Approach,” *IFAC Proceedings Volumes*, vol. 46, no. 19, pp. 167-171, 2013.
- [20] “Uppaal,” [Online]. Available: <http://www.uppaal.org/>. [Accessed 29 April 2018].
- [21] A. Anier, J. Vain and L. Tsiopoulos, “DTRON: a tool for distributed model-based testing of time critical applications,” *Proceedings of the Estonian Academy of Sciences*, vol. 66, no. 1, p. 75–88, 2017.
- [22] A. Anier, *DTRON tutorial*, <https://cs.ttu.ee/dtron/dtronTutorial.pdf>, 2018.
- [23] S. A. Jacklin, “Survey of Verification and Validation Techniques for Small Satellite Software Development,” in *Space Tech Expo Conference*, Long Beach, 2015.
- [24] L. S. Sterling, *The Art of Agent-Oriented Modeling*, London: The MIT Press, 2009.

Appendix 1 – Thread Identification Protocol

Date: April 17th 2018 and April 24th 2018

Participants: Academic Supervisor of the MCS team and 10 development team members

Process description: the participants were asked to suggest threads related to the MCS software. There were no right or wrong suggestions, all of them were accepted and documented in order to support a free, creative and informal ambiance.

Suggested threads:

No	Thread description
1	Even if different modules are successfully integrated in the test environment, problems with integration can still occur in the product environment.
2	Wrong software is accidentally sent up to the satellite and the satellite won't work anymore.
3	A malicious user gains access to MCS back office and send for example the shut-down command to the satellite.
4	An error in bit stuffing (for example a missing flag) causes radio silence.
5	The satellite sends false information during flyover.
6	A malicious user gains direct access to Active MQ and posts malicious messages directly.
7	Only one team member possesses crucial information like server passwords.
8	We are currently not monitoring our server and we might lose important data when the Orbit server fails.
9	An amateur might cause so much noise with his/her antenna that we can't hear our satellite anymore.
10	We might accidentally consider an amateur Telemetry as our own.
11	When we do not have clear description of what are the physical limits of our antenna, we might accidentally break it with our steering system (for example by ordering it to rotate more than it physically can).

No	Thread description
12	When something physically happens with the satellite in space and we do not have enough information for understanding the cause and consequences.
13	When we are not able to store data for the requested period of time.
14	An accidental drop table in the database (live or test).
15	The electronics on the satellite is damaged because of magnetic storms.
16	When someone accidentally deletes our Git repository and we will not be able to figure out after restoring it which version is on the satellite.
17	When we fail to send an urgent command to the satellite during planned flyover because of MCS performance issues.
18	There are several places where the message to/from the satellite can get lost.
19	When either Docker of the server on which Docker is running crashes.

Appendix 2 – Likelihood and Impact Definition Protocol

Date: May 4th 2018

Participants: Academic Supervisor of the MCS team and 10 development team members

Process description: the participants were asked to evaluate pre-defined likelihood and impact scores for Product Risks related to the MCS software and to request changes where necessary. The final likelihood and impact scores for threads are given in the table below.

ID	Category	Risk description	Likelihood	Impact
PR-1	Reliability risk	There may be integration issues between different modules and systems.	4	4
PR-2	Reliability risk	Incorrect or faulty software may be uploaded on the satellite.	4	5
PR-3	Security risk	A malicious user may gain access to MCS back office.	2	4
PR-4	Reliability risk	An error in bit stuffing may cause radio silence.	4	3
PR-6	Security risk	A malicious user may gain access to Active MQ.	2	4
PR-8	Reliability risk	We may lose data due to poor reaction of server failure.	4	3
PR-9	Reliability risk	Radio noise caused by amateurs may hinder our communication with the satellite.	1	4
PR-10	Reliability risk	Amateur Telemetry may accidentally be considered as our own by the MCS.	1	4
PR-11	Reliability risk	The MCS may cause physical damage to our antenna.	3	4
PR-12	Reliability risk	Physical damage to the satellite in space may not be identified, communicated or understood through the MCS.	4	4
PR-13	Reliability risk	Data may not be stored according to requirements and for the required amount of time.	2	2
PR-14	Security risk	Data may accidentally be deleted.	2	4
PR-15	Security risk	The Git Repository may accidentally be deleted.	1	3

ID	Category	Risk description	Likelihood	Impact
PR-16	Performance risk	MCS performance issues may cause delays in sending commands to the satellite.	2	4
PR-17	Performance risk	The message to/from the satellite or a part of it may be lost when transferring the message from one module or system to the other.	4	4
PR-18	Reliability risk	Docker or the server on which Docker is running may crash unexpectedly for several reasons.	3	2

**Appendix 3 – Testing and Verification Plan for Mission
Control Software (MCS)**

TUT Mektory Nanosatellite Programme

**Testing and Verification Plan for Mission Control Software
(MCS)**

Document versions

<i>Document ID and date</i>	<i>Author</i>	<i>Remarks</i>
<ul style="list-style-type: none">• v1 06.12.2016	<ul style="list-style-type: none">•	<ul style="list-style-type: none">• First version
<ul style="list-style-type: none">• V2 02.05.2018	<ul style="list-style-type: none">•	<ul style="list-style-type: none">• MATLAB removed• YouTrack changed to GitLab• Roles and responsibilities updated• Estimated launch date postponed

Contents

1. Introduction

2. Objective

3. Scope

4. References

5. Resources

6. Testing and verification strategy

6.1 Unit Testing

6.2 System and Integration Testing

6.3 Stress and Load Testing

6.4 User Acceptance Testing

6.5 Automated Regression Testing

7. Control procedures

8. Roles & responsibilities

9. Schedules

1. Introduction

The Mission Control Software (MCS) is the main interface between the ground operator and the satellite. It consists of six main components that are responsible for the following operations:

- Radio driver – communication between the Radio Control Unit, MCS and Active MQ;
- Two-line element (TLE) service – downloading the list of orbital elements of the satellite in a two-line element set (TLE) from web on an hourly basis (once an hour) and forwarding the information messages to Active MQ;
- Monitoring (Limit check);
- Public Telemetry Endpoint;
- Archiver – logging data received from Active MQ and sending the logs to database.

2. Objective

The objective of this test and verification plan is to describe the strategies, methods, environments and tools used in the test and verification process of the MCS.

The MCS shall correspond to Technology Readiness Level (TRL) 6 according to the ESA Technology Readiness Levels Handbook for Space Applications, which is considered the service level agreement in the current test and verification plan.

3. Scope

The scope includes verification of all components of the MCS and all possible interfaces between the MCS and external systems and applications. Software verification is the testing done to make sure the software code performs the functions it was designed to perform [22].

The scope does not include testing and verification of external systems and applications that are communicating with the MCS. All data received from external systems are taken

“as it is” and it is not validated, unless the validation of incoming data is a specified functionality of the MCS.

Integration testing is done by creating an environment that is as close to the estimated real situation as possible. Using real external systems (or their test environments) is preferred. If this is not possible, then emulators will be created for simulating external systems and applications.

4. References

- ESA Technology Readiness Levels Handbook for Space Applications

5. Resources

- GitLab – sprint planning, story/task descriptions, bug reporting;
- TestLink test management system;
- Jenkins and Selenium for automated system testing;
- Slack communication platform – team communication, real-time messaging;
- Git version control system.

6. Testing and verification strategy

The development process of the MCS follows an iterative model. Testing is a part of every phase in the cycle.

Different techniques and open source tools may be used for testing, based on the specific situation. Test cases are defined and automated, where reasonable and possible.

6.1 Unit Testing

The general approach in the project is that developers are responsible for creating unit tests for their own code using suitable tools.

More specific guidelines for creating and running unit tests as well as code coverage goals are to be defined in the next version of this document.

6.2 System and Integration Testing

Test cases for system and integration testing are planned and conducted based on use cases.

Priorities are set on use cases in order to define the most critical functionalities.

Different testing techniques are used for creating system test cases (boundary values, equivalence classes, classification trees etc.). The specific techniques are chosen based on concrete functionalities and priorities.

System and integration tests shall be automated mainly if they can be used as regression tests later. Otherwise they will be executed manually.

Integration testing may require creation of emulators for external systems.

System and integration testing will be planned and conducted by software tester(s).

6.3 Stress and Load Testing

The purpose of Stress Testing is to determine the robustness of software by testing beyond the limits of normal operation.

Stress tests will be designed with the aim to confirm that the system can maintain its functionality under heavy load and to find:

- Possible concurrency issues or deadlocks;
- Certain type of bugs that may be difficult to detect over the relatively short period of time when testing is performed.

6.4 User Acceptance Testing

The purpose of acceptance test is to confirm that the system is ready for operational use. Acceptance test cases will be conducted based on pre-defined use cases.

6.5 Automated Regression Testing

Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements.

Regression tests for most critical functionality can be designed and automated as soon as a component has reached the necessary level of maturity for carrying out its main function.

7. Control procedures

The main channel for problem reporting is GitLab.

Incidents encountered while testing that are verified as errors are reported as error or “bug” tickets in GitLab.

8. Roles & responsibilities

Product owner, team leader;

Architect/developer: Mission control software architecture and core;

Developer: Communication protocol;

Developer: Telemetry/payload commanding;

Developer: Integration with Mission Planning;

Developer: MCS user interface;

Developer: Mission planning;

Developer: Common Validation and Verification System;

Quality Assurance Specialist: Testing and verification.

9. Schedules

Sprint length: 1 week.

Project reviews: at the end of every semester (first review on December 7th 2016).

Estimated live/launch of the satellite: 2019.

Appendix 4 – Low-level test cases for MCS communication with EPS

Black-box techniques used for test case creation:

- Use case testing;
- Boundary value testing;
- Equivalence class testing.

Table 14 – Test Cases

ID	Test condition	Test case description	Expected result
TC-1	CO-1	Read from one register (No 25) / read the temperature of Battery	The temperature of Battery A in C
TC-2	CO-1	Read from two registers (No 25 and 26) / read the temperature and voltage of Battery A	The temperature (in C) and voltage (in V) of Battery A
TC-3	CO-1	Read from 0 registers	No response
TC-4	CO-1	Read from all registers	Data from all registers
TC-5	CO-2	Read from 102 registers	Wrong command (0x01)
TC-6	CO-2	Read from 256 registers	Wrong command (0x01)
TC-7	CO-2	Write to a read-only register	Wrong command (0x01)
TC-8	CO-1	Write to one register No 59	Data written
TC-9	CO-1	Write to registers 59 and 60	Data written
TC-10	CO-2	Write to all registers	Wrong command (0x01)
TC-11	CO-2	Write to register No 256	Wrong command (0x01)

ID	Test condition	Test case description	Expected result
TC-12	CO-4 CO-1	Send two concurrent commands: Write to one register No 59 Read from all registers	Data written
TC-13	CO-4 CO-2	Send two concurrent commands: Read from 256 registers Write to one register No 59	Wrong command (0x01)
TC-14	CO-3	Send a reset command to address 0x04	Wrong command (0x01)
TC-15	CO-3	Send a Programmable Interface Controller (PIC) Program Memory Update command to address 0x04	Wrong command (0x01)
TC-16	CO-3	Send a PIC Program Memory Read command to address 0x04	Wrong command (0x01)
TC-17	CO-3	Send a Read Memory Checksum command to address 0x04	Wrong command (0x01)
TC-18	CO-2	Send a read command to address 0x55	Wrong command (0x01)
TC-19	CO-2	Send a write command to address 0x55	Wrong command (0x01)
TC-20	CO-2	Send a write to multiple registers command to address 0x55	Wrong command (0x01)
TC-21	CO-3	Send a PIC Program Memory Update command to address 0x04	Wrong command (0x01)
TC-22	CO-3	Send a PIC Program Memory Read command to address 0x04	Wrong command (0x01)
TC-23	CO-3	Send a Read Memory Checksum command to address 0x04	Wrong command (0x01)
TC-24	CO-2	Send more than one byte with CTRL byte	No response: CTRL byte of the frame is ignored on reception
TC-25	CO-3	Send a command with CMD (0x01)	Wrong command (0x01)
TC-26	CO-3	Send a command with CMD (0x02)	Wrong command (0x01)
TC-27	CO-3	Send a command with CMD (0x04)	Wrong command (0x01)
TC-28	CO-3	Send a command with CMD (0x05)	Wrong command (0x01)
TC-29	CO-3	Send a command with CMD (0x07)	Wrong command (0x01)

ID	Test condition	Test case description	Expected result
TC-30	CO-3	Send a command with CMD (0x08)	Wrong command (0x01)
TC-31	CO-3	Send a command with CMD (0x09)	Wrong command (0x01)
TC-32	CO-3	Send a command with CMD (0x011)	Wrong command (0x01)
TC-33	CO-1	Write 2 bytes to a read-write register	Data written
TC-34	CO-2	Write 3 bytes to a read-write register	Wrong command (0x01)
TC-35	CO-5	Verify the storage of Ku-band high speed downlink frames received from EPS	Data is stored in ground station data storage in raw IQ data format in addition to normal decoded format for a period of 1 month
TC-36	CO-5	Verify the storage of data downlinked to the ground segment, via UHF and Ku-band downlink channels	Data is stored in a dedicated server for up to 1 years after the completion of the mission