Requirements Modeling Language and Automated Testing for CubeSats

Abdulaziz Alanazi
Department of Computer Science
North Dakota State University
Fargo, ND, USA 58102
abdulaziz.alanazi@ndsu.edu

Andrew B. Jones
Department of Computer Science
North Dakota State University
Fargo, ND, USA 58102
andrew.jones.4@ndsu.edu

Jeremy Straub
Department of Computer Science
North Dakota State University
Fargo, ND, USA 58102
jeremy.straub@ndsu.edu

***Abstract*—Developing cyber-physical systems such as CubeSats requires dynamic engagements of stakeholders, developers, tester, analysts, and project managers. With their different backgrounds, the technicality of analyzing functional and nonfunctional requirements may lead to miscommunication between groups. Thus, applying requirements modeling language (RML) into an engineering system development project will enhance the quality of requirements as well as ease the analysis and elicitation processes.**

In this paper, the importance of implementing RML into a project lifecycle of an engineering system (such as a CubeSat system) is described. A case study of using requirements mapping matrix (RMM) for requirements validation and verification of a typical CubeSat system is provided. This paper concludes with a summary of results and a discussion of future work.

Keywords—cubesat, cube satellite, RML, requirements modeling

I. Introduction

The analysis and elicitation of systems requirements is not an easy practice in any engineering environment. If requirements were not properly analyzed and documented, it may cause the project to fail. Although several methods and practices are implemented in the engineering systems development lifecycle, failure still exists mainly due to weak requirements analysis and elicitation practices. Requirements may be ambiguous, incomplete, or inconsistent. They might also be long, complex, and hard to comprehend.

Developing cyber-physical systems such as CubeSats requires dynamic engagements of stakeholders, developers, tester, analysts, and project managers. With their different backgrounds, the technicality of analyzing functional and nonfunctional requirements may lead to miscommunication between groups. Thus, applying requirements modeling language (RML) into an engineering system development project will enhance the quality of requirements as well as ease the analysis and elicitation processes.

Requirements Modeling Language (RML) is a language designed specifically to visually model requirements for easy consumption by stakeholders. RML provides management tools that help organize the phases of the entire project lifecycle. Although other models such as UML are commonly used in engineering systems development projects, they might present a logical structure with requirements and features of an engineering system clearly to a developer or a tester while being hard and complex to a stakeholder. Whereas RML provides objective models that can visually present functional and nonfunctional requirements for better analysis and elicitation activities. RML also helps CubeSat developers to group and prioritize their requirements to identify ambiguity or inconsistency easily.

In this paper, the importance of implementing RML into a project lifecycle of an engineering system (such as a CubeSat system) is described. A case study of using requirements mapping matrix (RMM) for requirements validation and verification of a typical CubeSat system is provided. This paper concludes with a summary of results and a discussion of future work.

II. Background

A. CubeSats

CubeSats are miniature satellites that are widely used for exploring space in low-Earth orbit (LEO): orbits with an altitude between 100–1200 miles (160–2000 km) [1]. The concept of the CubeSat was developed in 1999 by Bob Twiggs of Stanford University and Jordi Puig-Suari of California Polytechnic State University [2]. CubeSats can be used for different missions such as observing the atmosphere [3], detecting Gamma Rays [4], detecting magnetic fields [5] or tracking space debris [6].

CubeSats are popular for a number of reasons. These include having a relatively simple design, low cost, and compatibility with commercial-off-the-shelf parts (COTS) and open-source software components [7]. The low cost is relative to the cost of building a larger satellite - which can cost millions of dollars, while building a CubeSat can be done for thousands or tens of thousands of dollars [8]. The use of COTS components can aid developers in planning, designing, building and testing their CubeSats. Examples of COTS include mobile phone hardware, low cost cameras, consumer radiometers, and basic RF beacons [9].

Another advantage of CubeSats is that they provide simplified launch vehicle integration capabilities due to their standardized form factor. CubeSats commonly range in size from the 1U form factor (approximately 10 cm x 10 cm x 10 cm with a mass of around 1.33 kg) to larger sizes such as the 2U (20 cm x 10 cm x 10 cm) and 6U (30 cm x 20 cm x 10 cm) variants [10]. Although this is not a complete list of available form factors.

The CubeSat general requirements are discussed in [11]. CubeSat system requirements describe what services the system

978-1-7281-2832-0/19/$31.00 ©2019 IEEE

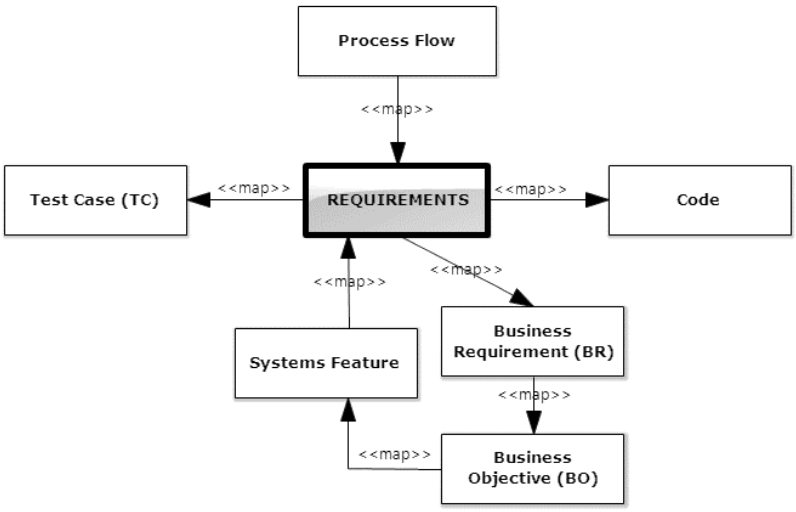

Fig. 1. Diagram depicting RMM mapping levels.

should provide, and the quality of the service necessary to meet technical, financial, educational and other goals [10]. The system requirements may include mission objectives, interface, functional and nonfunctional, and physical requirements.

III. Requirements Modeling Language

Requirements Modeling Language (RML) is a collection of diagrams used to model software from the business analysis or product management perspective [12]. RML is concerned with a project's goals and objectives, instead of complex system design models (which is the focus of UML and SysML) [13]. Moreover, RML models form boundaries based on different sections of the system, which are intended to bound the problem space [14]. RML models fall into four categories: objective models, system models, people models, and data models [13].

A. Requirements Mapping Matrix (RMM)

A Requirements Mapping Matrix (RMM) is best used by a requirement management model tool which can automate the process of checking for missing links [12]. It is intended to ease the process of stakeholders' consumption when it comes to understanding and analyzing the requirements. RMM is an RML objective model, which means it is an application that can visually present the systems requirements and business rules. RMM contains multiple levels of mapping that map process flow steps to requirements, requirements to business rules, business objectives to features, features to requirements, requirements to code, and requirements to test cases, as illustrated in Fig. 1.

RMM supports the fundamental limitation of human brain theory proposed by [15], which proposed that a human brain can only remember seven plus or minus two tasks at once. Thus, RMM only includes eight or less columns in its matrix. The first three columns (L1, L2, L3) are process flows, and the rest of the matrix includes business objectives, requirements, business rules, code and test cases.

One of the main purposes of the RMM is to ensure that the current product being developed is on the right track [12]. Furthermore, it aids in determining whether extra unspecified functionalities are added, which may affect the scope of the project. Moreover, it is intended to ensure that the project progresses as per the desired direction and that every requirement is tested thoroughly.

RMM may help reduce the number of "shall statements" which may help non-technical stakeholders to comprehend it more readily (due to reduction of overall content). RMM is similar to the Requirement Traceability Matrix (RTM), which is another RML objective model. The dynamics of both models rely on the process of documenting the links between the user requirements (proposed by the client) to the system being built [12]. It works as a high-level documentation, which maps and traces user requirements with the corresponding test cases. This is intended to ensure that for each and every requirement there is an adequate level of testing successfully fulfilled.

The aspect that differentiates Requirements Traceability Matrix (RTM) and Requirements Mapping Matrix (RMM) is that RTM supports many to many relationships, where each requirement can map to many test cases (and a test case maps to many requirements). In contrast, RMM supports the one-to-many relationship where each cell shows the relationship of a specific requirement with a particular test [12].

RMM is initiated by reviewing all the test cases defined for each requirement. It enables the determination of which requirement has the most frequent defects during the testing process. The focus of any testing engagement is, and should be, maximum test coverage. By coverage, it simply means that there is a need to test everything there is to be tested. The aim of any testing project should be end-to-end test coverage (if feasible). RMM may aid in determining the number of test cases for each requirement. RMM helps to link the requirements, test cases, and defects accurately. Furthermore, RMM may enhance the quality of the application as all the features are tested. It also assures that the "Quality Control" can be achieved as software gets tested for unforeseen scenarios with minimal defects and all

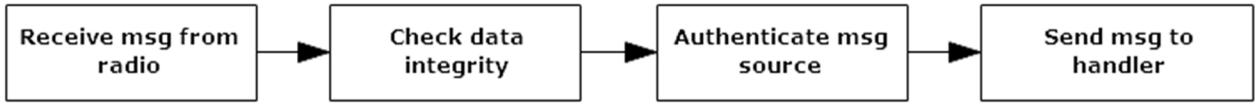

Fig. 2. Diagram of process flow 1.

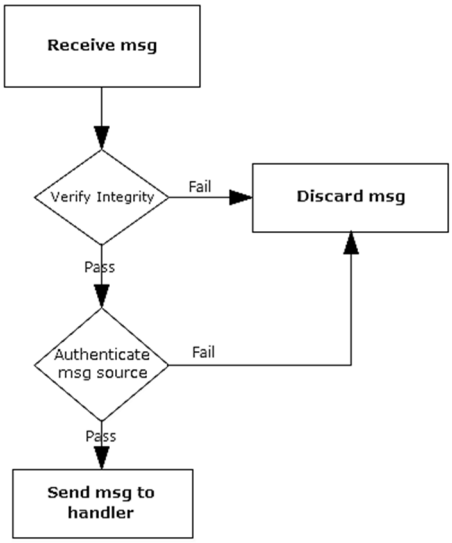

Fig. 3 Diagram of process flow 2.

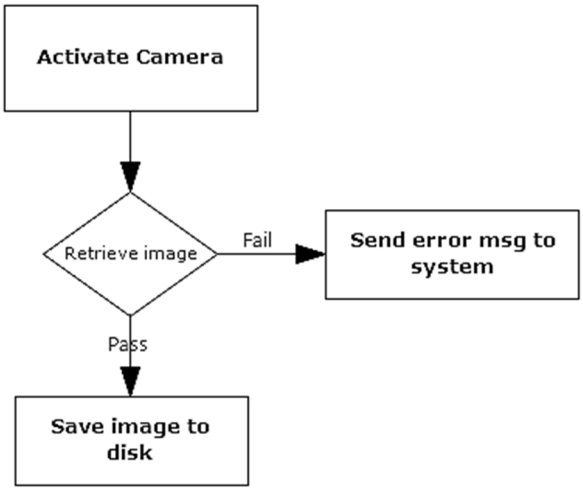

Fig. 4 Diagram of process flow 3.

functional, non-functional requirements, and business rules are being met [12]. RMM is intended to reduce the testing time for software applications, ensure the scope of the project is well determined, that its implementation is achieved as per the customer requirements and needs, and that the cost of the project is controlled.

IV. CubeSat Requirements Modeling

Developing cyber-physical systems such as CubeSats requires dynamic engagements of stakeholders, developers, tester, analysts, and project managers. With their different backgrounds, the technicality of analyzing functional and nonfunctional requirements may lead to miscommunication between groups. Thus, applying requirements modeling language (RML) into an engineering system development project may enhance the quality of requirements as well as ease the analysis and elicitation processes.

RMM may help CubeSat developers and systems analysts prioritize requirements, identify missing requirements, and highlight unnecessary requirements. CubeSat system requirements may include a list of shall statements for functional, non-functional or business requirements, such as:

- FR.1. System shall measure the inclination angle to the sun within (10-18) degrees.
- FR.2. System shall measure the temperature of the spacecraft within (0.2-3.0) degrees Celsius.
- FR.3 System shall confirm antennas have deployed successfully.

Business Objective (BO)	L2 Process Step	Requirement	Business Rule (BR)	Code	Test Case ID	Test	Result
Increase system security	Ensure data integrity	system must verify data integrity	system can detect errors and receive msg	msg.validate()	TC.0101	validate	Pass
Increase system security	Ensure authentication of msg	system must authenticate received msg	system can validate msg source	msg.authenticate()	TC.0102	authenticate	Pass
Business Objective (BO)	L3 Process Step	Requirement	Business Rule (BR)	Code	Test Case ID	Test	Result
Increase data retention	Save image to file	system must be able to initialize camera	system can activate camera	take.img()	TC.0201	retrieve	Pass
Increase data retention	Save image to file	system must be able to save image to disk	system can store image to file	save.img()	TC.0202	save	Pass

Fig. 5 RMM for CubeSat system with tests mapped to requirements.

Figs. 2-4 illustrate process flows L1, L2, L3 respectively. The corresponding RMM matrix with the code and tests mapped to requirements for the NDSAT (example satellite concept) CubeSat Radio and Camera systems is depicted in Fig. 5.

A sample CubeSat system requirements document is presented to show one of the main components of the CubeSat system, On-board Computer (OBC). The OBC software design shall use 60% of the available system memory. The following section will discuss the functional and nonfunctional requirements specifications for the CubeSat OBC.

Test Case ID	TC.0201			
Test Case Description	Test the OBC's ability to save images taken by the camera			
Test Scenario	Have the OBC retrieve an image from the camera and save it to disk			
Test Case Result	Pass			
Preconditions:	[A] Camera is powered; [B] OBC is connected to camera; [C] OBC has disk space available			
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Step Result**
1	Activate/Initialize Camera	Camera should initialize	As Expected	Pass
2	Retrieve Image from Camera	Image taken from camera is set to variable	As Expected	Pass
3	Save image to disk	Image saved as a file on disk	As Expected	Pass

Test Case ID	TC.0202			
Test Case Description	Test the OBC's ability to save images taken by the camera			
Test Scenario	Have the OBC retrieve an image from the camera and save it to disk			
Test Case Result	Pass			
Preconditions:	[A] Camera is powered; [B] OBC is connected to camera; **[C] OBC has no disk space available**			
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Step Result**
1	Activate/Initialize Camera	Camera should initialize.	As Expected	Pass
2	Retrieve Image from Camera	Image taken from camera is set to variable.	As Expected	Pass
3	Save image to disk	Image is not saved to disk. Error displayed.	As Expected	Pass

Test Case ID	TC.0101			
Test Case Description	Receive radio transmission from base station			
Test Scenario	Radio message is received, parsed for transmission errors, and authenticated			
Test Case Result	Pass			
Preconditions:	[A] Radio is powered; [B] Radio is connected to OBC; [C] Data integrity maintained; [D] Properly authenticated			
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Step Result**
1	Retrieve message from Radio	Message should be retrieved	As Expected	Pass
2	Evaluate checksum	Checksum should be accurate	As Expected	Pass
3	Authenticate message	Message should be flagged as from a valid source	As Expected	Pass

Test Case ID	TC.0102			
Test Case Description	Receive radio transmission from base station			
Test Scenario	Radio message is received, parsed for transmission errors, and authenticated			
Test Case Result	Pass			
Preconditions:	[A] Radio is powered; [B] Radio is connected to OBC; **[C] Data integrity not maintained**; [D] Properly authenticated			
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Step Result**
1	Retrieve message from Radio	Message should be retrieved	As Expected	Pass
2	Evaluate checksum	Checksum inaccuracies should be caught	As Expected	Pass

Fig 6. CubeSat test cases for receiving radio messages and for saving images to disk.

V. Software Design Document

A. Introduction

OBC is the central component of the CubeSat system. It is connecting all subsystem components, so it can control subsystems by allowing subsystems to connect with the communication & Data handling with the Electrical Power System (EPS).

1) Purpose

This document describes the NDSAT Test Satellite on OBC software architectural design.

2) Scope

This initial document is designed to show the OBC functional and non-functional requirements for the NDSAT.

B. OBC Requirements

1) Functional Requirements (FR)

FR	Action
FR001	OBC must run within 0.5 second upon deployment
FR002	OBC must initiate antenna signal upon deployment
FR003	OBC must send and receive images from ground station
FR004	OBC must be able to control and configure onboard camera
FR005	OBC must be able to reprogram itself

2) Non-Functional Requirements (NFR)

NFR	Action
NFR001	Software architecture must include software tools that support debugging. (Maintainability)
NFR002	Software architecture must include Version Control System (VCS) in repository. (Reliability)
NFR003	Test cases with results must be documented (Testability)
NFR004	System shall operate in space for at least 6 months (Availability)
NFR005	OBC must be able to operate in space magnetic field (Operability)

C. Scenarios and Test Cases

The sample test cases are depicted in Fig. 6. After executing the test cases if any defects are found that too can be listed and mapped with the business requirements, test scenarios and test cases. For instance, If TS1.TC1 fails i.e. Capture image option though enabled does not work properly then a defect can be logged. If the defect ID auto-generated or manually assigned number is D01, then this can be mapped with BR1, TS1, and TS1.TC1 numbers.

VI. Conclusion and Future Work

In this paper, the importance of implementing RML into a project lifecycle of a CubeSat system was discussed. A case study of using requirements mapping matrix (RMM) for requirements validation and verification of a typical CubeSat system was provided. Future work will include verifying and characterizing the efficacy of the proposed approach, testing it in an actual project and updating it based on this experience.

Acknowledgment

Facilities, materials, and equipment used for this work have been supplied by the North Dakota State University (NDSU) Department of Computer Science and the NDSU College of Science and Math.

References

[1] A. Alanazi and J. Straub, "Engineering Methodology for Student-Driven CubeSats," *Aerospace*, vol. 6, no. 5, p. 54, May 2019.

[2] A. Chin, R. Coelho, R. Nugent, R. Munakata, and J. Puig-Suari, "CubeSat: The Pico-Satellite Standard for Research and Education," in *AIAA SPACE 2008 Conference & Exposition*, 2008.

[3] D. Selva and D. Krejci, "A survey and assessment of the capabilities of Cubesats for Earth observation," *Acta Astronaut.*, vol. 74, pp. 50–68, 2012.

[4] V. Dániel *et al.*, "Terrestrial gamma-ray flashes monitor demonstrator on CubeSat," in *CubeSats and NanoSats for Remote Sensing*, 2016, vol. 9978, p. 99780D.

[5] I. Garrick-Bethell *et al.*, "Lunar magnetic field measurements with a cubesat," in *Sensors and Systems for Space Applications VI*, 2013, vol. 8739, p. 873903.

[6] R. Lucken, N. Hubert, and D. Giolito, "Systematic space debris collection using Cubesat constellation," in *7th European Conference for Aeronautics And Aerospace Sciences (EUCASS) Systematic*, 2017.

[7] A. Alanazi and J. Straub, "Statistical Analysis of CubeSat Mission Failure," in *Proceedings of the 2018 AIAA/USU SmallSat Conference*, 2018, pp. 4–9.

[8] S. Sassi, "Power generation and solar panels for an MSU CubeSat," Mississippi State University, 2016.

[9] L. Zea, V. Ayerdi, S. Argueta, and A. Munoz, "A Merthodology for CubeSat Mission Selection," *J. Small Satell.*, vol. 5, no. 3, pp. 483–511, 2016.

[10] H. Reza, C. Korvald, J. Straub, J. Hubber, N. Alexander, and A. Chawla, "Toward requirements engineering of cyber-physical systems: Modeling CubeSat," in *2016 IEEE Aerospace Conference*, 2016, vol. 2016-June, pp. 1–13.

[11] A. Mehrparvar, "Cubesat design specification," *CubeSat Program, Calif. Polytech. State Univ.*, 2014.

[12] J. Beatty and A. Chen, *Visual Models for Software Requirements*.

Microsoft Press, 2012.

[13] C. Adams, "What is RML (Requirements Modeling Language)?" [Online]. Available: https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/3270/What-is-RML-Requirements-Modeling-Language.aspx. [Accessed: 05-May-2019].

[14] A. Chen, "Comparing UML vs. RML for Complete, Full Coverage Business Analysis." [Online]. Available: https://seilevel.com/requirements/comparing-uml-vs-rml-for-complete-full-coverage-business-analysis. [Accessed: 05-May-2019].

[15] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, no. 2, pp. 81–97, 1956.