aerospace

MDPI

Article

Engineering Methodology for Student-Driven CubeSats

Abdulaziz Alanazi and Jeremy Straub *

Department of Computer Science, North Dakota State University, 1320 Albrecht Blvd., Room 258, Fargo, ND 58102, USA; abdulaziz.alanazi@ndsu.edu or jastraub@gmail.com

* Correspondence: jeremy.straub@ndsu.edu; Tel.: +1-701-231-8196

Received: 31 December 2018; Accepted: 25 March 2019; Published: 13 May 2019

Abstract: CubeSats are widely used by universities and research institutions all over the world. Their popularity is generally attributed to the use of low-cost components, free student labor and simple design. They have been shown to encourage Science, Technology, Engineering and Math (STEM) students to become involved in designing, implementing and testing a real functioning spacecraft system. Projects like this encourage students from different disciplines to team up to design and build CubeSats, providing interdisciplinary work experience. Participating students vary in their expertise in developing such systems. Some will work on the project for years while others are not willing to spend two or three consecutive semesters developing a CubeSat project. Despite their simplicity in design and low cost, CubeSats are considered great engineering systems for exploring space. Nevertheless, a large number of CubeSat projects fail due to having an unclear mission, ambiguous system requirements and a lack of documentation. Students need to have a clear vision of how to build a real CubeSat system that can be launched and that can function in space. Thus, this paper proposes engineering methodologies and tools to help students develop CubeSat systems. These tools can help students with planning, collecting, eliciting and documenting the requirements in a well-defined manner. This paper focuses on student-driven CubeSat projects designed by students and faculty members. Additionally, data is presented in this paper to identify the challenges and needs of CubeSat developers. Plans for future work are also discussed.

Keywords: CubeSat; student satellite projects; system engineering; software engineering; COTS; STEM

1. Introduction

The idea of designing a small cube-shaped spacecraft was initiated by Bob Twigg and Jordi Puig-Suari in 1999 [1]. CubeSats were initially designed for educational purposes: to help students to become acquainted with the space environment [2] Swartwout termed these university-class CubeSats [3]. Now, CubeSats are used not only for educational purposes but also for scientific, governmental and commercial purposes. Thus, the number of launched CubeSats has dramatically increased in the last eight years and is still expected to grow. Figure 1 shows the number of launched CubeSats in 2018 and what types of CubeSats are most commonly used. It shows that 325 one-unit (1U) CubeSats were launched from 166 educational institutions in 47 countries [4]. Despite this high demand for CubeSat development, the failure of CubeSat systems is high. This is due to several reasons including the tools used, the models and students' levels of experience. Due to the use of internal or self-generated requirements, stemming from an educational mission or for other reasons, some universities and research institutions may predominately focus on the system and component level design while neglecting the full elicitation of requirements needed beforehand.

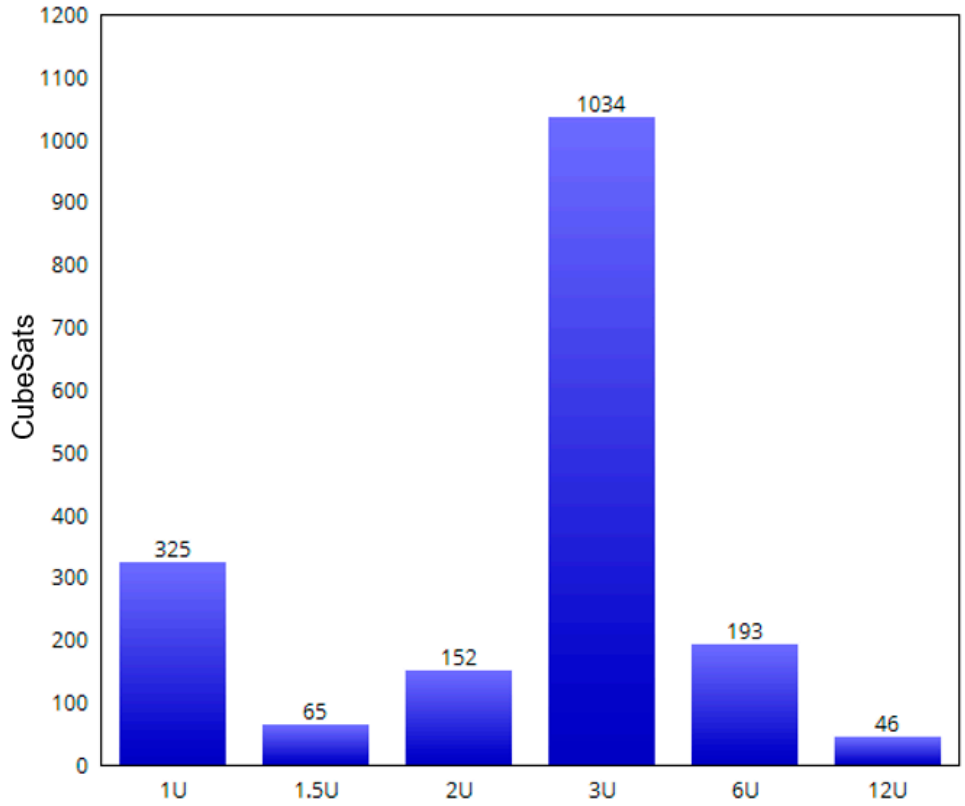

Figure 1. The number of launched CubeSats (based on [4]): Shows the number of launched CubeSats and their types between 2003–2018. The figure includes all CubeSat missions including, government, commercial and educational.

This paper is designed to help students who are currently developing or will develop their first student-driven CubeSat system. It outlines the tools, methodologies and lessons learned from projects that have been developed by other universities. It also presents a CubeSat reference model to help schools meet their mission requirements. This model includes different tools that help in achieving verification and validation. Such tools include modeling, simulation, integration and analytical models. This paper further presents a system engineering methodology that supports rapid development and helps students save on time and budget for their project.

2. Background

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). The idea of the CubeSat was developed in 1999 by Bob Twiggs of Stanford University and Jordi Puig-Suari of California Polytechnic State University [1]. These small satellites were initially designed for university education purposes: to teach college students how to design and build small systems that can efficiently be used for space education [4]. One-unit CubeSats with simple missions are mostly used by universities who are developing CubeSat projects for the first time. Despite their limited size, 1U CubeSats are capable of collecting data and communicating significant amounts of data to ground stations using commercial off-the-shelf (COTS) components, including phone cameras, radiometers or beacons [5]. In the past five years, however, universities have started developing more 3U and larger CubeSats. These missions typically include an advanced scientific goal in their mission design [6]. Figure 2 depicts a typical 1U CubeSat.

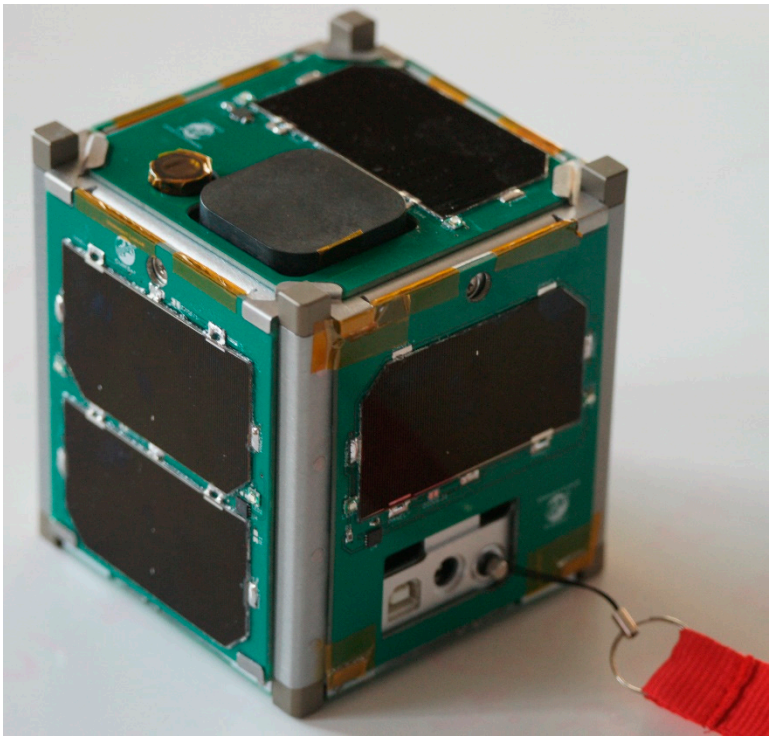

Figure 2. A typical 1U CubeSat [7].

CubeSats can be used for different missions such as observing the atmosphere [8], detecting Gamma Rays [9], detecting magnetic fields [10] or tracking space debris [11]. What makes CubeSats so popular is their simple design, using open source hardware and software components. Using COTS components helps developers plan, design, build and test their CubeSats in a short development process. From 2003 to October 2018, 878 university-class CubeSats have been launched [4].

2.1. CubeSat Design and Developing Processes

It is well-known that decisions made in the concept phase of a program can determine approximately 70% of the cost of a program [12]. The concept phase is where the CubeSat mission and goals are identified. This stage, however, may be neglected by some university students who are willing to accept higher risks of mission failure. Even with a simple CubeSat mission, students initiate their CubeSat design process by listing all the COTS components needed for the CubeSat subsystems. This includes the flight computer, communication components and sensors [13].

Having flexible hardware and software systems for their CubeSat is the major aim for many developers. This flexibility in major systems components can reduce the amount of time and budget spent on designing the CubeSat. Thus, many universities follow the CubeSat design specification documents when designing their CubeSat system [14] as well as vendor-supplied reference models. Additionally, developers should also consider designing the software architecture carefully. The primary goal is to design a robust and extensible piece of software. When designing a software system, an effective command set must be provided and major software components must pass environment and operational testing [15]. Nearly all satellites have at least one flight computer devoted to performing tasks related to communications and data handling. Many satellites have additional computers devoted to the operation of specific subsystems or tasks such as attitude control or their scientific or other payload. This allows for the simple delegation of operations and may provide a redundancy in the event of computer failure [16].

2.2. University Demand for CubeSats

Universities and research institutions heavily rely on open-source software and hardware components when designing their CubeSats. These components are defined by two well-known organizations that set the definitions, regulations and license aspects of such open source systems. The Open Source Hardware Association (OSHA) defines open-source hardware as any hardware of which the design is made publicly available so that anyone can study, modify, distribute, make and sell a design or hardware based on that design. The hardware's source, the design from which it is made, is available in the preferred format for making modifications to it [17]. The Open Source Initiative (OSI) defines open-source software as software that can be freely used, changed and shared (in modified or unmodified forms) by anyone. Many people make open-source software and distribute it under licenses that comply with the Open Source Definition [18].

There is a high demand for CubeSats by universities due to their simple design, low cost and inexpensive commercial-of-the-shelf components (COTS). Excluding launch costs, the components for a traditional small satellite with a simple mission could significantly exceed $50,000 and require an average of 7–10 years of developmental time, whereas building a student-driven 1U CubeSat with a simple mission may cost less than $5000, with an average of less than two years of development time [12,19]. CubeSats cost less because universities can use COTS and open-source software and hardware parts donated from vendors. Additionally, the cost of labor is relatively inexpensive compared to professional engineers at space agencies and commercial developers. Student-driven CubeSats are developed by college students with different majors including aerospace, mechanical, electrical and software engineering and computer science.

Another reason for CubeSats' popularity is that they can be a great source for collecting data from space once they are in orbit. Despite its small size, a CubeSat subsystem can handle significant communication tasks such as the transmission of telemetry data to a ground station and the receipt of commands [16]. Between 2003 and 2018, the total number of launched university-class CubeSats reached 878 [4]. This number is expected to grow as the number of countries who show an interest in this technology is rapidly increasing. However, the high demand for CubeSat systems does not guarantee the success of their missions. Statistics show that, for all university-class missions between 1994 and 2017, roughly 40% of CubeSats failed to meet their mission objective [5]. Reasons for CubeSat failure include an early loss or a launch failure or may remain unknown. Figure 3 shows the CubeSat mission failure rate from 1994 to 2017.

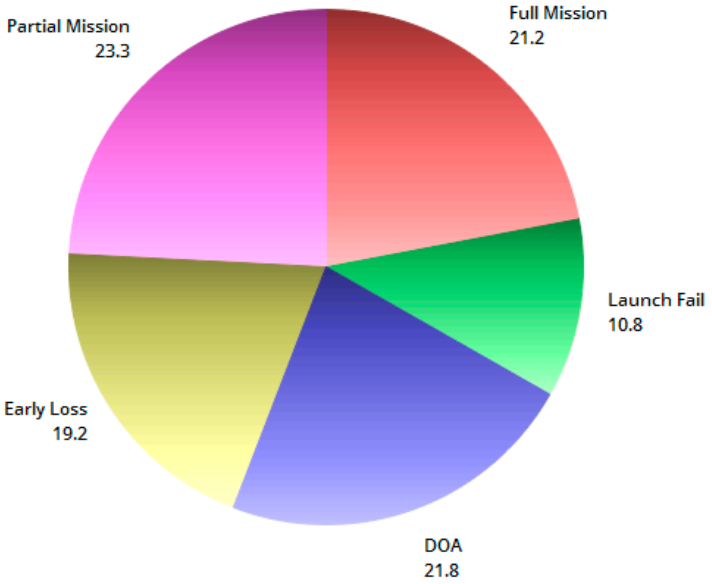

Figure 3. The CubeSat mission status from 1994 to 2017 (created based on [5]).

3. Investigating Reasons for Failure

This paper only focuses on the reasons of failure of the early stages of student-driven CubeSat projects (i.e., before launching). Reasons of failure vary between CubeSats, depending on their mission purposes. The reasons may include but are not limited to the following:

- Exceeding development time: where the approximate development time is 1–2 years, some projects may exceed the time because the participants are inexperienced or graduate and leave the project.
- Limited functionality of components: the cost of space-grade components may be expensive for CubeSat projects. For instance, the average cost of GaAs solar cells is $3000 [20]. Thus, students may, instead, choose less expensive COTS with a limited functionality for their project.
- Lack of system testing: this may be due to the lack of availability of testing tools. The testing time may affect the project life-cycle.
- Requirements analysis: misunderstanding and communication issues between student developers and stakeholders (i.e., principal investigator, developers and testers)
- Inexperienced team: students learn while working on the project through trial and error. Some may be participating in the project only to receive an extra credit. Some may graduate and leave school before completing the project [21].
- Lack of documentation: students may not have proper documentation for their project.

In order to ensure university-class CubeSat mission success and to reduce the rate of failure, a systems engineering methodology can play a significant role. Nevertheless, students may implement formal methodologies that suit traditional satellites but not CubeSats. For instance, CubeSats have more severe constraints such as size, weight and power as compared to traditional satellites. A system engineering methodology will not only help students drive their CubeSat project's success but also ensure the validation and verification of CubeSat mission success. However, CubeSat developers' needs and challenges need to be investigated to identify the factors that are associated directly with CubeSat mission success and failure.

A recent study of CubeSat mission failures pinpointed some factors associated with the failure of CubeSat missions [22]. However, there are more than seven factors that are associated with CubeSat mission success or failure. The listed factors were selected and identified based on the initial focus group of CubeSat developers. The following factors were coded for statistical analysis as follows:

1. Testing time reduction (variable name: TTR): coded as 0 if it did not occur and as 1 if it occurred.
2. Design problems (variable name: DesPr): coded as 1 if the problems were related to tools, 2 if the problems were related to the models and 3 if the problems were related to both.
3. Availability of model for modification (variable name: Mod): coded as 0 if not available and as 1 if available.
4. Ease of addition or deletion of components (variable name: AddDel): coded as 1 if easy and as 2 if difficult.
5. System design objectives met (variable name: SysMet): coded as 0 if not met and as 1 if they were.
6. Mission objectives met (variable name: MMet): coded as 0 if not met and as 1 if they were.
7. Whether one model was employed as a reference model for different missions (variable name: Mission): coded as 0 if not employed and as 1 if it was.

A list of all the associated factors with their variable names and codes is included in Table 1.

Table 1. The factors associated with CubeSat success/failure.

Factor	Variable Name	Code
testing time reduction	TTR	occurred (1), did not occur (0)
designing problems	DesPr	tools (1), models (2), both (3)
availability of modifications	Mod	yes (1), no (0)
adding/deleting components	AddDel	easy (1), difficult (0)
system design objectives	SysMet	yes (1), no (0)
mission objectives	MMet	yes (1), no (0)
using a reference model	Mission	yes (1), no (0)

A statistical analysis of these independent variables associated with the success or failure of CubeSats is described subsequently. The participants in this study were students who have developed or are currently working on CubeSat projects at their universities. The principal objective of this study is to identify and assess factors associated with reducing the rate of CubeSat failure through the application of a system engineering approach.

The survey was given to 120 individuals identified at the CalPoly CubeSat Workshop; however, only 35 were fully completed. Approximately 48% of respondents reported experiencing tools failure, such as communication problems with simplex and duplex radios, very high frequency (VHF) or ultra-high frequency (UHF) transceivers failure and power failure. To avoid an interruption while transmitting signals, satellites need full-duplex system radios. However, some groups included transceivers in their CubeSat system that use half-duplex radio—especially groups that included mobile devices into their system.

Twenty-four percent of respondents thought that the major challenges for them were caused by the models they used. This included integration and analytical models (excluding flight software models). Additionally, 28% stated that the problems they had were due to both the tools and models. Of those responding, 57% reported that they had not considered techniques for reducing the system testing time requirements, either because the testing was performed externally by vendors or because they had never used methods that could facilitate a reduction in testing time. Required testing included the testing of both hardware and software systems including thermal, radiation and vacuum testing. The other 43% respondents reported enjoying a reduction in testing time through the use of qualification testing and requirements analysis.

Model-based systems engineering (MBSE) has been proposed as a reference model and methodology to help schools meet their mission requirements. MBSE includes different tools that facilitate the realization of verification and validation. Such tools include modeling, simulation, integration and analytical models. However, only 35% of respondents reported following one reference model for different CubeSat projects, while 65% had never considered using one [22].

The factors listed in Table 1 are considered independent variables (IVs) contributed toward CubeSat mission success or failure. A CubeSat's mission success constitutes a dependent variable (DV). All the IVs discussed above can be treated as being nominally independent of each other; they are not hierarchically related. Therefore, in order to ascertain whether each one of the IVs is related to another, chi-square tests of independence would be appropriate with mission success as the DV and one IV for each test of independence. In addition, it is possible that two or more of the IVs can jointly explain the success or failure of the mission, and for this, a logistic or binomial regression model can be constructed with mission success as the DV and all the IVs.

The chi-square tests of independence were conducted first. These tests were performed with two assumptions: Each variable should be categorical and independent of each other, and each level of each variable should have an expected value of at least 5. Tables 2–8 present the statistical analyses for the factors associated with the success/failure of CubeSat missions.

Table 2. The chi-square test for testing time reduction.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	1.020	1	0.313		
Continuity Correction	0.431	1	0.512		
Likelihood Ratio	1.018	1	0.313		
Fisher's Exact Test				0.481	0.255
Linear-by-Linear Association	0.991	1	0.32		
N of Valid Cases	35				

Table 3. The chi-square test for designing problems.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	0.326	2	0.849	1.000	
Likelihood Ratio	0.333	2	0.847	0.904	
Fisher's Exact Test	0.432			1.000	
Linear-by-Linear Association	0.007	1	0.931	1.000	0.546
N of Valid Cases	35				

Table 4. The chi-square test for the availability of modifications.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	1.225	1	0.268	0.541	0.306
Continuity Correction	0.232	1	0.630		
Likelihood Ratio	1.177	1	0.278	0.541	0.306
Fisher's Exact Test				0.541	0.306
Linear-by-Linear Association	1.190	1	0.275	0.541	0.306
N of Valid Cases	35				

Table 5. The chi-square test for adding/deleting components.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	0.122	1	0.726	1.000	0.525
Continuity Correction	0.000	1	1.000		
Likelihood Ratio	0.121	1	0.728	1.000	0.525
Fisher's Exact Test				1.000	0.525
Linear-by-Linear Association	0.119	1	0.730	1.000	0.525
N of Valid Cases	35				

Table 6. The chi-square test for the system design objectives.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	3.512	1	0.061	0.079	0.067
Continuity Correction	2.267	1	0.132		
Likelihood Ratio	3.477	1	0.062	0.139	0.067
Fisher's Exact Test				0.079	0.067
Linear-by-Linear Association	3.412	1	0.065	0.079	0.067
N of Valid Cases	35				

Table 7. The chi-square test for the mission objectives.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	0.015	1	0.901	1.000	0.591
Continuity Correction	0.000	1	1.000		
Likelihood Ratio	0.015	1	0.901	1.000	0.591
Fisher's Exact Test				1.000	0.591
Linear-by-Linear Association	0.015	1	0.903	1.000	0.591
N of Valid Cases	35				

Table 8. The chi-square test for using a reference model.

	Value	df	Asymptotic Significance (2-Sided)	Exact Sig. (2-Sided)	Exact Sig. (1-Sided)
Pearson Chi-Square	0.015	1	0.901	1.000	0.591
Continuity Correction	0.000	1	1		
Likelihood Ratio	0.015	1	0.901	1.000	0.591
Fisher's Exact Test				1.000	0.591
Linear-by-Linear Association	0.015	1	0.903	1.000	0.591
N of Valid Cases	35				

It can be observed that the p value of the Pearson Chi-Squared (χ^2) test statistic is 0.313, which is higher than the 5% significance level ($p = 0.313$). Therefore, no significant relationship between TTR and mission success or failure can be asserted.

A Fisher's exact test statistic would be more appropriate since some cells had expected counts of greater than 5. It can be observed that the test does not assert a significance since the p-value is much higher than the 5% significance level ($p = 1.000$). Therefore, no significant relationship between the design problems faced and mission success and failure can be asserted.

It can be observed that the Fisher's exact test statistic does not indicate a significance due to the p-value being higher than 0.05 ($p = 0.541$). This indicates that no significant relationship between the availability of modifications and mission success and failure can be asserted.

It can be observed that the Fisher's exact test statistic does not indicate a significance due to the p-value being higher than 0.05 ($p = 1.000$). This indicates that no significant relationship between the ease of addition or deletion of components and mission success and failure can be asserted.

As shown in the table, the Pearson chi-square test statistic as well as the Fisher's exact test statistic were significant at the 10% level, with p-values less than 0.1 ($p = 0.06$ and $p = 0.079$, respectively). This indicates that there was a significant relationship between the system objectives met and mission success and failure.

The chi-square test was not significant at the $p < 0.10$ level ($p = 0.9$). This indicates a lack of demonstration of a significant relationship between mission objectives being fulfilled and mission success/failure.

It can be observed that no significant relationship is demonstrated between the use of one reference model and mission success/failure ($p = 0.9$). The chi-square tests, therefore, demonstrated that only system objectives being met formed a critical factor (at the 10% significance level) that contributed to mission success. While the above tests indicated whether each factor contributed individually to mission success, their combined impact was also be investigated, in combinations and all together [22]. The prior study shows that system design objectives being met has a significant relationship with the CubeSat success/failure mission. The majority of students (60%) indicated that they think that they have not met their CubeSat mission objectives. This is due to satellite power or launch failure or to no communications being received from their satellite.

On the other hand, 38% of students believed that they had met their CubeSat mission objectives [22]. Despite the limited set of survey responses, the results indicate that, in general, successful missions have a lower testing time compared to failed missions. The data also shows that successful missions have less issues with tools comparing the implementation to models. Additionally, such missions have less models available for modification. Also, it was observed that the addition/deletion of components was less applicable to successful missions and that the ratio of mission objectives met to those not met was higher for such missions. Also, a reference model was used more in successful missions than in failed ones.

Additionally, other challenges such as a lack of experience as a CubeSat developer, the size of the team and the project budgeting/scheduling may also exist in university-class CubeSat projects. There are also constraints in university-class CubeSat designs that limit the functionality and performance of the CubeSat system. For example, size, mass and power are dominant constraints. Additionally, the cost and testing processes of hardware components are also significant constraints. Labor, materials, environmental testing and travel are also common costs of university CubeSat projects [13]. Students may be mainly concerned with the tools or models that lead to project failure, whereas project scopes, goals, methodology and standards may never be discussed among the project team. Thus, a further investigation is needed to analyze these problems. However, it may be easier to track CubeSat mission success than to track mission failure. Some repository databases of SmallSat systems (including CubeSats) can help CubeSat researchers with statistical figures and valuable data. These databases may include Swartwout's CubeSat Database [23], DK3WN SatBlog [24] and Erik Kulu's Database [4].

4. Student-Driven CubeSat Practices

Universities implement different methods and concepts into their CubeSat projects depending on their desired mission. Some universities apply tools and models to enhance the design process of their CubeSats; others may provide workshops to their STEM students before developing the actual CubeSat. In general, university students from different disciplines divide up into small teams to work on designing and developing their CubeSat project. For example, aerospace students may work on environmental analyses and control systems, mechanical engineering students might work on the structures and layout mechanisms, computer science students can test and develop software for data communication, and electrical engineering students can design and test the electrical power subsystem (EPS) and other electrical systems.

Universities apply different engineering models and tools for CubeSat system development. For instance, the University of North Dakota used model-based systems engineering tools to simplify and expedite requirements elicitation and specifications. They designed quality-attribute (QA) scenarios to document the nonfunctional requirements (NFR) of the CubeSat [25]. Figure 4 is an example of a scenario of availability usage.

With this tool, once a project has been created, specific attribute scenarios can be added. Each of these attributes must fall under a specific category. These categories are general quality attributes which are decided upon by the project lead (such as availability and maintainability). At California Polytechnic State University, students used the consortium requirements engineering (CoRE) method for their PolySat project [26,27]. Implementing the CoRE method can help in providing descriptions of the acceptable software behaviors. It also provides a clear understanding of the software requirements by using familiar terms and measured quantities. The visual representation of the CoRE behavioral model shows the relationship between the variables derived from the requirements (REQ) and variables from the environment or nature (NAT). The CoRE behavioral model is illustrated in Figure 5.

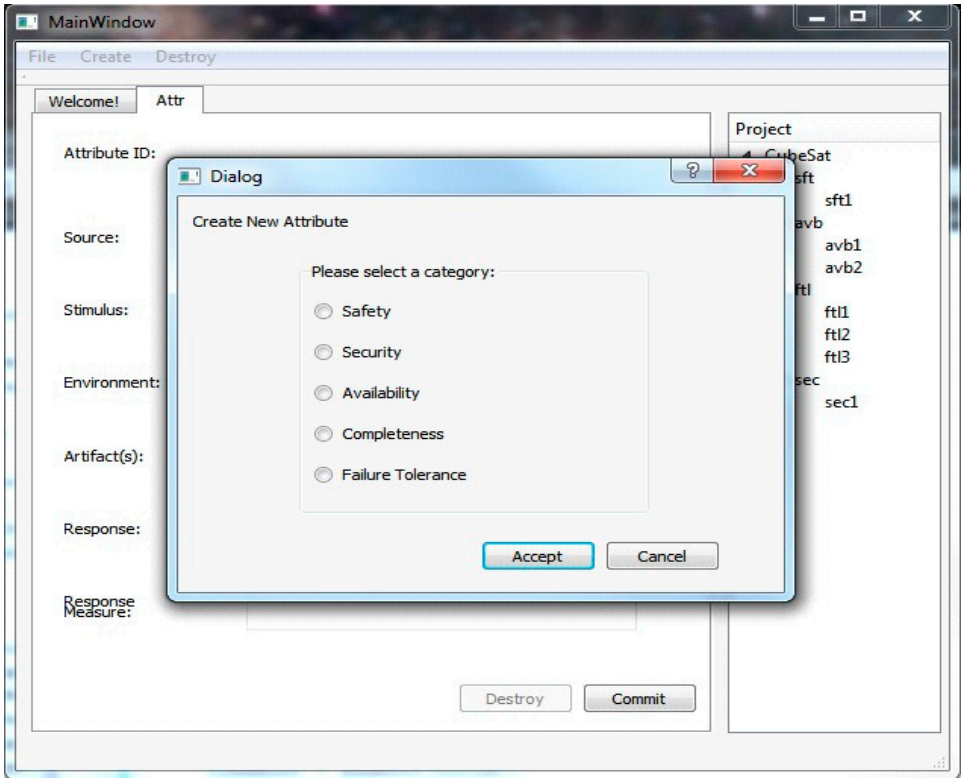

Figure 4. The quality attribute categories [25].

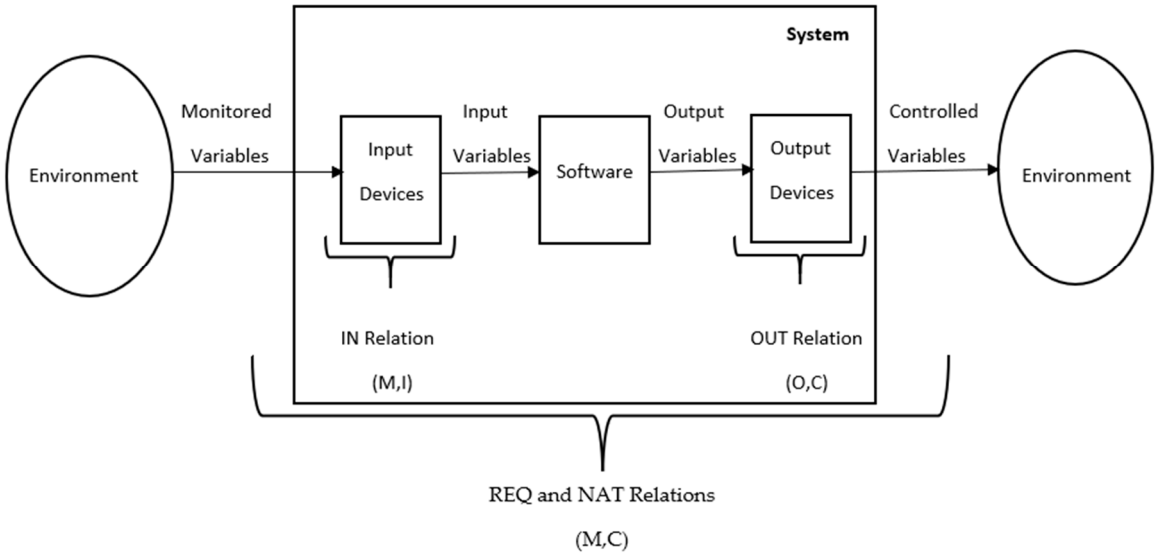

Figure 5. The consortium requirements engineering (CoRE) behavioral model, (modified from [26,27]).

The Technical University of Munich team, on the other hand, designed 3-D printed prototypes for better representations of their CubeSat and to aid in recruiting team members. The team implemented a 3-D printed CubeSat (MOVE-II) at an early stage to help students validate and verify CAD models to avoid potential design deficiencies [28]. Additionally, students at Universidad del Valle de Guatemala investigated the risk factors that might lead to a CubeSat failure. They designed a tool (risk matrix) to mitigate risk factors during the design phase of their project [29].

5. Engineering Methodology for University-Class CubeSats

In order to have an efficient development process, implementing a systems engineering methodology for a university's CubeSat project is necessary. Systems engineering is an interdisciplinary approach and a means to enable the realization of successful systems [30,31]. Students form into teams of developers, testers and designers with different backgrounds, desires and levels of experience. A systems engineering model developed by the Space Systems Working Group (SSWG) serves as an excellent starting point. The SSWG is a group of NASA staff, engineers, consultants, professors and students. This group developed a CubeSat systems reference model (CSRM) to guide students with steps and processes throughout their CubeSat project life-cycle. The CSRM model is a representation of a logical architecture design of a standard CubeSat system (see Figure 6). SSWG defined the CubeSat Reference Model CSRM as SysML model elements that can be populated to specify the logical architecture and design of a CubeSat's space and ground systems. The logical architecture decomposes the system into components that interact to satisfy system requirements. The components are abstractions of the physical components that perform system functionality but without imposing implementation constraints. CSRM applications and nomenclature were developed based on the roles and responsibilities of stakeholders (listed in Figure 6). The initial process has been initiated by the Object Management Group (OMG) to provide the CSRM as an OMG normative specification. For this, it must be SysML-compliant and implementation-free. It is a template model that provides building blocks that can be specialized to support MBSE CubeSat design.

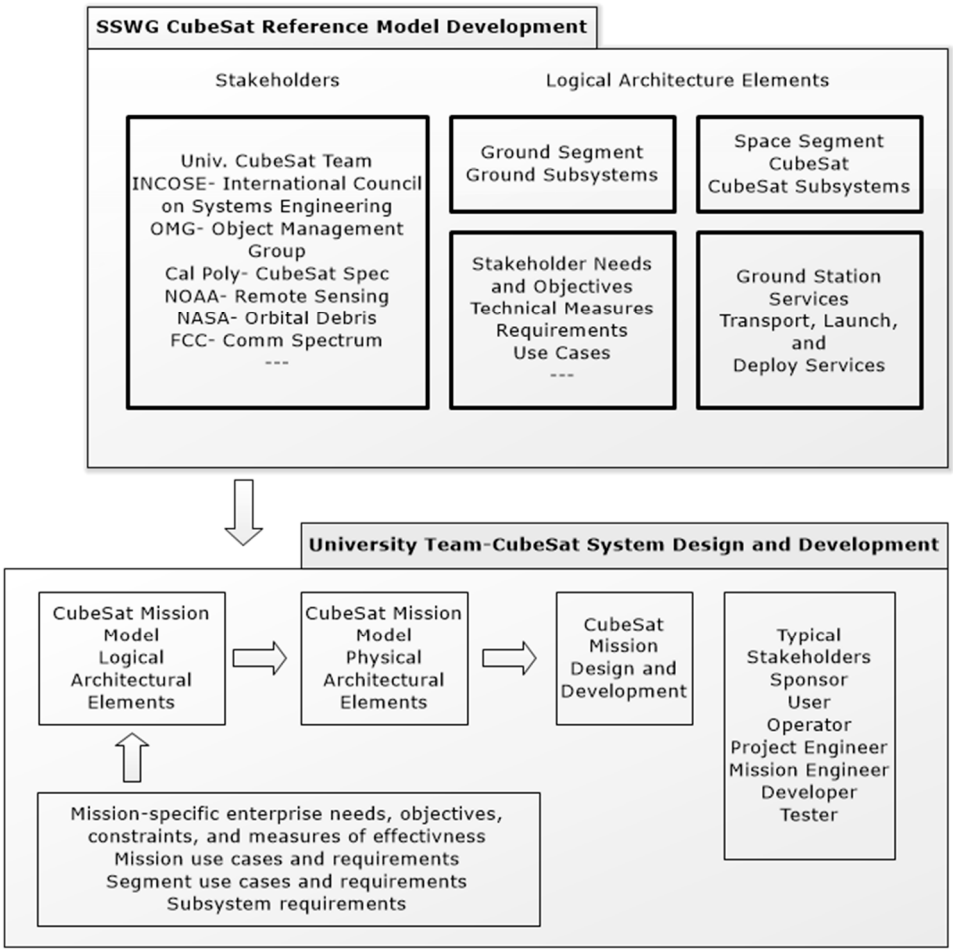

Figure 6. The Space Systems Working Group (SSWG)-CubeSat reference model (based on [32,33]).

For a project to develop its reference model based on CSRM, which in turn would form the basis for its mission-specific CubeSats, developers need to import the CSRM XML file into a graphical modeling tool. Then, they can update the model to accommodate the project stakeholder needs relative to project phases, standards, engineering processes and design artifacts. The stakeholder accommodation can be used to transition stakeholders from traditional systems engineering and document-based design to MBSE. Next, an independent review team will validate that the CRM complies with standard SysML modeling guidelines. OMG is responsible for establishing the CRM as a specification. OMG review and approval of the CRM will validate that the CRM is qualified to be a specification. The SSWG group is hoping that their CSRM model can help students with the challenges and needs that might occur during the developing processes, including addressing requirements, design, analysis, and validation and verification [32,33].

A university-class CubeSat project's life-cycle will include multiple phases and steps. However, some of these steps may be less critical than others and can be skipped or can receive less time for development. Figure 7 shows a sequential approach of a university-class CubeSat project life-cycle [31]. The agile methodology can be used for an experimental 1U CubeSat project. With this approach, the phases of a project life-cycle can be divided into small iterative steps and design testing scenarios so that they can detect errors at an early stage of the project and fix them quickly. Nevertheless, some students may still prefer to implement traditional methodologies, such as the v-model or waterfall methodologies for their CubeSat projects [34]. Others may also consider the spiral-model when they think of risk analysis as a significant factor [29].

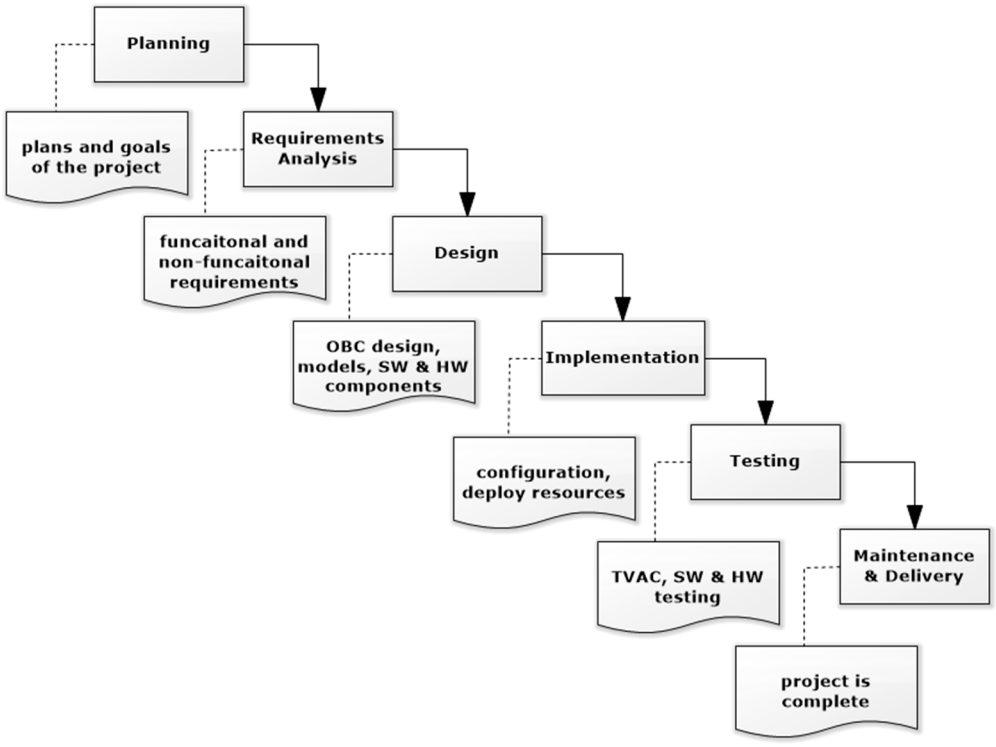

Figure 7. A university-class CubeSat project life-cycle (based on [31]).

The time required for each phase will vary. The concept phase, for instance, may take up to six months, whereas developing and testing the ground station may take a whole year. Figure 8 shows a notional timeline for project phases [13]. Some phases can, and typically do, run concurrently.

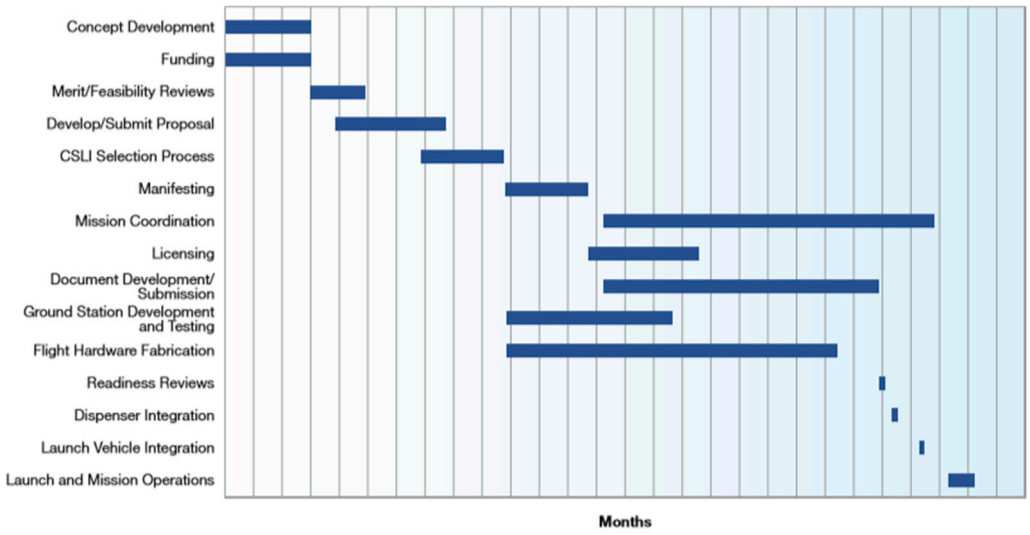

Figure 8. A timeline of a CubeSat project's phases [13].

A university-class CubeSat contains different subsystems such as an onboard computer (OBC), communication, EPS, attitude determination and control system (ADCS), data handling and payload. A block diagram of the CubeSat subsystem components is depicted in Figure 9. Each subsystem may be designed and developed separately by students before being combined with all of the components. Based on the priority of system requirements, the software developers, for instance, can take one requirement, code and test it and then work on the next requirement. Similarly, when it comes to designing the software for the system, the software team can divide up this process into small iterations instead of building all the system requirements at once (see Figure 10). Small iterations, each with four to five steps, can provide flexibility to the team to think freely and to detect errors at early stages. It can also help stakeholders to identify and add more requirements into the development process easily.

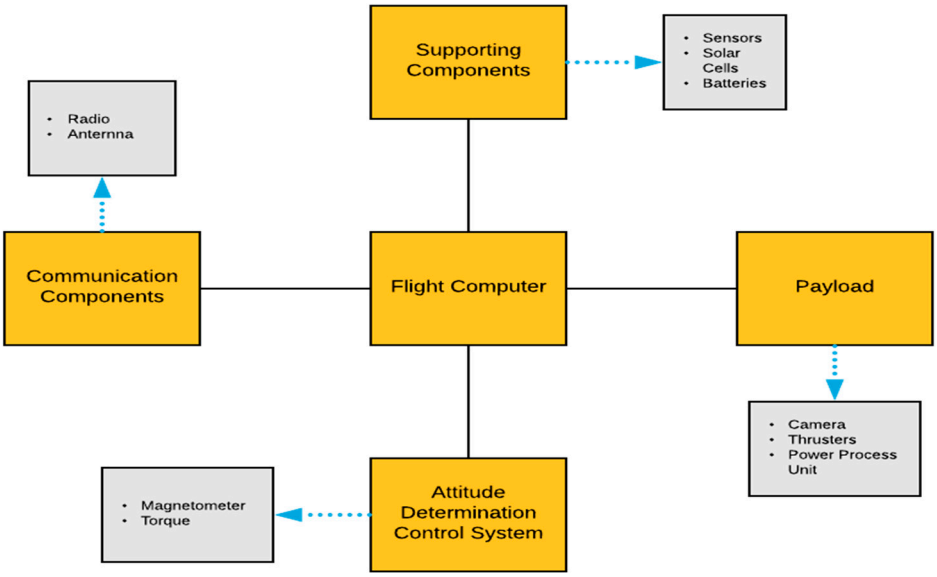

Figure 9. A block diagram of a CubeSat subsystem's components.

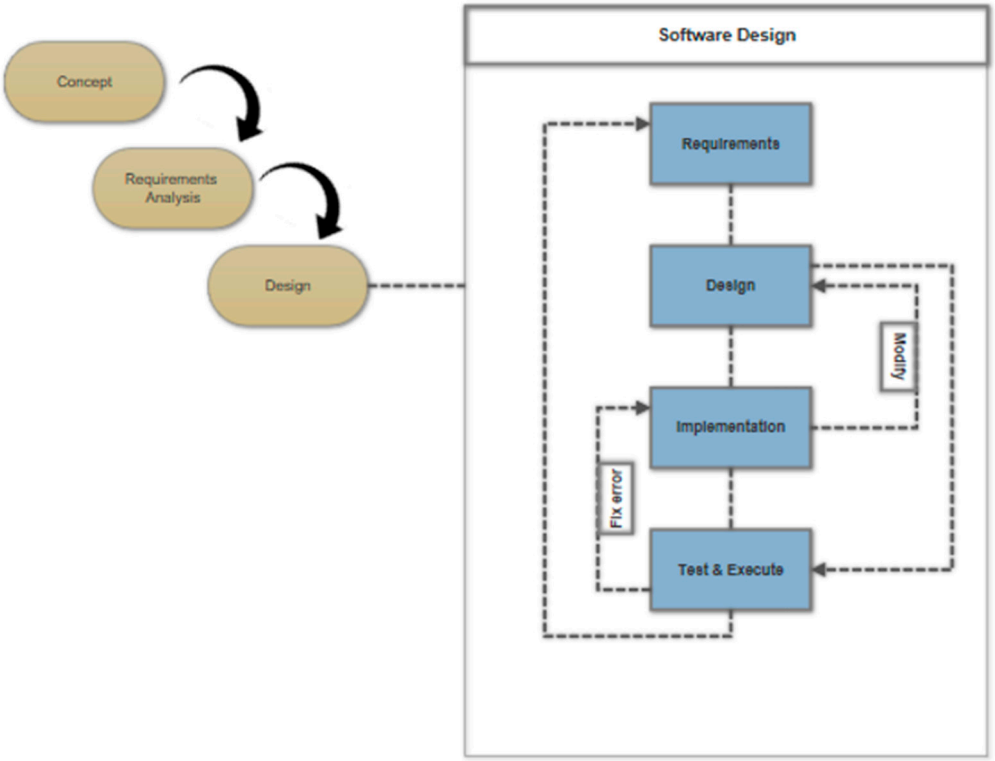

Figure 10. An iteration cycle in the software design phase [31].

6. CubeSat Software Requirements

It is highly desirable to create code that is reusable and fault-tolerant for both CubeSats and ground stations [35]. Typically, the team of students will need to develop software that can send and receive data, read sensors and operate mission payload if the COTS software is not available for some or all of these functions. To do this, the team must first set their CubeSat mission objectives and their goals for the project. To do this, in the requirements analysis, they need to gather as much information from stakeholders as they can and to document all of the needed specifications for the software they are designing. Next, during the design phase, the team must select the programing language and data storage and handling mechanism they are going to use in addition to implementing a selected design methodology to specify all aspects of the software system. Then, of course, they must build the actual code and verify that the software meets all given specifications.

With the limited budgets of CubeSat projects developed by university students, developers are not able to use hardware components with a high intelligence, where components can handle large operations and independently adjust when orbiting in space. Instead, components are directly connected to the flight computer to provide functionality and to manage operations, as illustrated in Figure 9.

Students must make numerous decisions during the design phase of the project. The team must analyze the requirements and develop testing scenarios. The team also needs to analyze and elicit requirements to make sure they are consistent and unambiguous. For instance, the following is an example of a functional requirement:

F-R: The system shall measure the temperature of the spacecraft within a specific range of −40 to 40 °C.

Testing scenarios must be created based on the given requirements. Assuming the testing team is following black box techniques, where the primary goal is to make sure that the system is functioning correctly without dealing with the low-level design (white box testing), the testers now play the role of end users. This means the testers test the application with input data then compares the output data with the expected values. According to the requirement above, if the temperature field only accepts

values between −40 and 40 °C, then—except any midrange special cases that must be assessed for other reasons—the tester only needs to design three test cases, one valid class and two invalid ones as follows:

- Valid: a number is between −40 and 40
- Invalid:
    - o a number is greater than or equal to 41
    - o the number is less than or equal to −41

Reducing the number of test cases, can also reduce the time required for testing and can help the developers to follow their iteration cycles and to swiftly jump to implementing the next requirement. Failed test cases are sent to the developers to fix. On the other hand, simple function calls such as PowerOn(), PowerOff() or GetTemp(), can be used for higher level software "black box" testing. The black box can be used to hide the details of the requirements where testers do not need to know the internal structural design. Since most students are willing to use open-source software and hardware, with simple mission objectives, they may be more interested in focusing on the behavioral structure of the system. They can do this by providing inputs and test and by comparing it to outputs.

7. Conclusions and Future Work

Lessons learned from university CubeSat projects are valuable for students who are newly exposed to this field. CubeSat mission success can be easily tracked and documented, to students' benefits, whereas tracking a mission failure is not as important. This research, though, has focused on investigating reasons for a mission failure from students' perspectives. Their challenges and needs provided insights into framing a CubeSat engineering methodology and designing the CubeSat project phases to ease the development process in student-driven CubeSats. The primary purpose of this project is to begin to create a software system engineering methodology for university-class CubeSat projects. This engineering methodology is designed to be used as a reference model for university projects. Thus, it is hoped that future projects will save time and cost instead of having to build one from scratch.

The challenges and needs of student developers need further investigation so that this methodology can be as valuable of a tool in the CubeSat project life-cycle. The university-class CubeSat life cycle can include different phases and steps divided into small iterations for rapid development. In many cases, the expected lifespan of a university CubeSat project is less than two years. In this study, however, only the first phases of the CubeSat projects are covered.

The main focus of this work was on how the CubeSat is developed according to systems requirements before launch. Thus, a further investigation of previous and current university CubeSat projects is needed. Such investigations can help in identifying the primary objective of university CubeSat designs and the tools and models, different systems aspects of structure, behavior, requirements and parameters they use. Future work will also include a statistical analysis that will measure the predicted success or failure of a CubeSat design, using systems complexity metrics.

Author Contributions: Conceptualization, J.S. and A.A.; methodology, A.A.; formal analysis, A.A.; resources, J.S.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, J.S.; supervision, J.S.; project administration, A.A. and J.S.

Funding: This research received no external funding.

Acknowledgments: Thanks are given to David Kaslow for his valuable feedback and suggestions on this research. Thanks are also given to Hassan Alsuhabi from the NDSU Department of Statistics for his assistance with the statistical analysis presented. A.A. is supported by the Saudi Arabian Cultural Mission and Saudi Arabia Northern Border University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chin, A.; Coelho, R.; Nugent, R.; Munakata, R.; Puig-Suari, J. Cubesat: The pico-satellite standard for research and education. In Proceedings of the AIAA Space 2008 Conference & Exposition, San Diego, CA, USA, 9–11 September 2008.
2. Erik, E. Nanosats Database [online Database]. Available online: https://www.nanosats.eu/ (accessed on 10 August 2018).
3. Swartwout, M. University-class satellites: From marginal utility to 'disruptive' research platforms. In Proceedings of the 18th Annual Conference on Small Satellites, Logan, UT, USA, 9–12 August 2004.
4. Robert, B. Distributed Electrical Power System in CubeSat Applications. Master's Thesis, Master of Science in Electrical Engineering, Utah State University, Logan, UT, USA, December 2011.
5. Swartwout, M. Reliving 24 Years in the Next 12 Minutes: A Statistical and Personal History of University-Class Satellites. In Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites, Logan, UT, USA, 4–9 August 2018. SSC18-WKVIII-03.
6. Xiaozhou, Y. The Status of University Nanosatellites in China. In Proceedings of the 14th Annual CubeSat Workshop, San Luis Obispo, CA, USA, 26–28 April 2017.
7. National Aeronautics and Space Administration. CubeSat Picture. Available online: https://www.nasa.gov/press-release/nasa-sets-coverage-schedule-for-cubesat-launch-events (accessed on 12 May 2019).
8. Selva, D.; Krejci, D. A survey and assessment of the capabilities of CubeSats for Earth observation. *Acta Astronaut.* **2012**, 74. [CrossRef]
9. Dániel, V.; Pína, L.; Inneman, A.; Zadražil, V.; Báča, T.; Platkevič, M.; Stehlíková, V.; Nentvich, O.; Urban, M. Terrestrial gamma-ray flashes monitor demonstrator on CubeSat. *Cubesats Nanosats Remote Sens.* **2016**. [CrossRef]
10. Garrick-Bethell, I.; Lin, R.P.; Sanchez, H.; Jaroux, B.A.; Bester, M.; Brown, P.; Cosgrove, D.; Dougherty, M.K.; Halekas, J.S.; Hemingway, D.; et al. Lunar magnetic field measurements with a CubeSat. *Sens. Syst. Space Appl.* **2013**, *8739*, 873903. [CrossRef]
11. Lucken, R.; Hubert, N.; Giolito, D. Systematic space debris collection using CubeSat constellation. In Proceedings of the 7th European Conference for Aeronautics and Aerospace Sciences (EUCASS), Milan, Italy, 3–6 July 2017. [CrossRef]
12. Fortescue, P.; Swinerd, G.; Stark, J. CubeSat Structures. In *Spacecraft Systems Engineering*, 4th ed.; John Wiley and Sons Inc.: Hoboken, NJ, USA, 2011; pp. 540–542, ISBN 9780470750124.
13. CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers. Available online: https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf (accessed on 20 October 2018).
14. Twiggs, B.; Puig-Suari, J. *CUBESAT Design Specifications Document*; Stanford University and California Polytechnical Institute: San Luis Obispo, CA, USA, August 2003.
15. Zea, L.; Ayerdi, V.; Argueta, S.; Muñoz, A. A Methodology for CubeSat Mission Selection. *J. Small Satell.* **2016**, *5*, 483–511.
16. Dabrowski, M.J. The Design of a Software System for a Small Space Satellite. Master's Thesis, Graduate College, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 2005.
17. The Open Source Hardware Association (OSHA). Available online: https://www.oshwa.org (accessed on 15 October 2018).
18. The Open Source Initiative (OSI). Available online: https://www.opensource.org (accessed on 15 October 2018).
19. Straub, J.; Whalen, D. Evaluation of the Educational Impact of Participation Time in a Small Spacecraft Development Program. *Educ. Sci.* **2014**, *4*, 141–154. [CrossRef]
20. List of CubeSat Components Prices. Available online: https://www.cubesatshop.com/ (accessed on 15 October 2018).
21. Straub, J. Extending the student qualitative undertaking involvement risk model. *J. Aerosp. Technol. Manag.* **2014**, *6*, 333–352. [CrossRef]
22. Alanazi, A.; Straub, J. Statistical Analysis of CubeSat Mission Failure. In Proceedings of the 2018 AIAA/USU SmallSat Conference, Logan, UT, USA, 4–9 August 2018.
23. Swartwout, M. Statistical Figures. Available online: https://sites.google.com/a/slu.edu/swartwout/home/cubesat-database (accessed on 30 October 2018).

24. Open-Source CubeSat Database Repository. Available online: http://www.dk3wn.info/p/ (accessed on 30 October 2018).
25. Reza, H.; Rashmi, S.; Alexander, N.; Straub, J. Toward Model-Based Requirement Engineering Tool Support. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2017. [CrossRef]
26. Weiz, N. Analysis of Verification and Validation Techniques for Educational CubeSat Programs. Master's Thesis, Master of Science in Computer Science, California Polytechnic State University, San Luis Obispo, CA, USA, May 2018.
27. Ward, P.T.; Mellor, S.J. *Structured Development for Real-Time Systems*; Prentice Hall Professional Technical Reference: Upper Saddle River, NJ, USA, 1991.
28. Weisgerber, M.; Langer, M.; Schummer, F.; Steinkirchner, K. Risk Reduction and Process Acceleration for Small Spacecraft Assembly and Testing by Rapid Prototyping. In Proceedings of the German Aerospace Congress, Munich, Germany, 5–7 September 2017.
29. González, D.; Rodríguez, D.; Birnie, J.; Bagur, J.; Paz, R.; Miranda, E.; Solórzano, F.; Esquit, C.; Gallegos, J.; Álvarez, E.; et al. Guatemala's Remote Sensing CubeSat—Tools and Approaches to Increase the Probability of Mission Success. In Proceedings of the 2018 AIAA/USU SmallSat Conference, Logan, UT, USA, 4–9 August 2018.
30. The International Council on Systems Engineering. Systems Engineering Definition. Available online: https://www.incose.org (accessed on 5 November 2018).
31. Alanazi, A. Methodology and Tools for Reducing CubeSat Mission Failure. In Proceedings of the 2018 AIAA Space and Astronautics Forum and Exposition, AIAA SPACE Forum, Orlando, FL, USA, 17–19 September 2018. AIAA 2018-5122.
32. Kaslow, D.; Louise, A.; Asundi, S.; Bradley, A.; Curtis, I.; Bungo, S.; Robert, R. Developing a CubeSat Model-Based System Engineering (MBSE) Reference Model-interim status. In Proceedings of the IEEE Aerospace Conference 2017, Big Sky, MT, USA, 7–14 March 2015. [CrossRef]
33. Hammond, W.E. *Space Transportation: A Systems Approach to Design and Analysis*; AIAA: Reston, VA, USA, 1999.
34. Aerosp, J.; Manag, T.; Campos, S. Design of a Nanosatellite Ground Monitoring and Control Software—A Case Study. *J. Aerosp. Technol. Manag.* **2016**, *8*, 211–231.
35. Manyak, G. Fault Tolerant and Flexible CubeSat Software Architecture; Master's Thesis, Master of Science in Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA, USA, June 2011.