

A Case Study to Develop Systems Engineering Tools for Enhancing Reliability and Efficiency of CubeSat Design and Development

By Ryohei TAKAHASHI,¹⁾ Masahiro FUJIWARA,¹⁾ Shunichiro NOMURA,¹⁾ Taiki KARAKAWA,¹⁾ Shingo NISHIMOTO,¹⁾ Keidai Iiyama,¹⁾ Kanta YANAGIDA,¹⁾ Akihiro ISHIKAWA,¹⁾ Shuhei MATSUSHITA,¹⁾ Ryu FUNASE,¹⁾ Shinichi NAKASUKA¹⁾

¹⁾ *The University of Tokyo, Japan*

(Received June 21st, 2021)

In recent years, CubeSats have been used to build satellite constellations and for scientific exploration, and the number of satellites and the number of companies designing and manufacturing them has been increasing. With the increase in the number of satellites and companies, not only the functions of CubeSats to be developed but also the process of designing and developing CubeSats and the tools used in the process have been developed. On the other hand, there are not many university satellite projects that have developed the design and development process of satellites themselves though a large number of university satellites have been launched and failed due to reliability problems. To solve the problem, a project is conducted to develop tools to enhance the reliability and efficiency of university satellite projects in a new 6U CubeSat project. In the project, requirements are extracted from past CubeSat projects, and adequate methods are selected to fulfill the requirements. This paper introduces the overview of the project and summarizes lessons learned of development and operations.

Key Words: CubeSat, Systems Engineering, Concurrent Engineering, Reliability, Efficiency

1. Introduction

In recent years, the industry, especially in the United States, has been refining the design and development process of CubeSats. Planet, an operator of a constellation of 3U CubeSats uses sophisticated development and verification processes where they have built a Hardware in the Loop simulator that can be connected to multiple satellites simultaneously to simulate a constellation.¹⁾ The simulator can be remotely accessed and execute tests 24 hours a day to enable the efficient development of software for the satellites and the ground stations. Spire has developed and launched 3U and 6U CubeSats for radio and meteorological observations in an efficient manner where they have developed its original web system that stores the development status of each satellite, the components used, software versions, and test results.²⁾ This makes it possible to quickly trace the results of development. In addition, Spire has introduced its original scheduling system, which reduces the risk of delays in a satellite assembly, increases the opportunity for design and software upgrades, and reduces the amount of component inventory. Team Xc at NASA Jet Propulsion Laboratory completes the conceptual design of one CubeSat in one or two weeks and performs about 20~30 conceptual designs per year.³⁾ To carry out such a fast conceptual design, Team Xc has developed an original concurrent engineering tool that connects experts in multiple locations in real-time to carry out the design simultaneously. While the development process for CubeSats has been refined in the industry, there are few examples of such refinement in university satellites, presumably due to the following reasons.

- Difficulty in establishing a refined satellite development process because student developers cannot stay in the project for a long time due to graduations.

- Limited budgets and lack of resources to develop original tools.

On the other hand, more sophisticated design and development processes are essential for university satellites to carry out more advanced missions and to achieve research results certainly. To address this issue, this study describes the activities of the University of Tokyo in defining the requirements for improving the design and development process for university satellites, and formulating the process and tools for improvement, and describes the lessons learned in this process.

The chapters are organized as follows. First, the demand and requirements for the tool in the university satellite are shown in Chapter 2. Based on the results of the requirements, the concept of operation of the tool is presented in Chapter 3. Chapter 4 presents the implementation strategy of the tool to meet the requirements. Chapter 5 presents the results of the trade-offs and implementations of the specific implementations that make up the architecture. Chapter 6 presents the results of the construction and Chapter 7 summarizes the lessons learned.

2. Requirement Extraction

This chapter describes the requirements for the upgrading of the satellite design and development process. This study aims to improve the efficiency and reliability of the design and development process of CubeSat, based on the efforts of Planet, Spire, and others mentioned in the introduction. In this paper, the requirements for efficiency and reliability of CubeSat design and development processes are defined based on the lessons learned of 6U CubeSat EQUULEUS which has been designed and developed by the University of Tokyo.⁴⁾

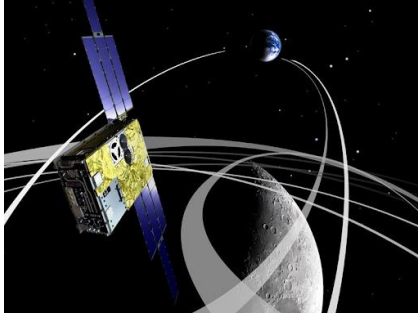


Fig. 1. Image of EQUULEUS mission ⁴⁾

2.1. Objective of Improvement

The objectives of improving the design and development process can be summarized as follows

- Improve efficiency
- Enhance reliability

Efficiency can be improved by eliminating the wasteful work involved in running the process and reducing the human resources required. It has a side effect that engineers can allocate their time more into the part which requires creativity and can produce research novelty.

Reliability can be enhanced by preventing human errors such as miscommunication, reducing rework during testing, and improving the success rate of satellite missions. It enables the universities to execute more complex and advanced missions successfully.

2.2. EQUULEUS

EQUULEUS is a deep space exploration CubeSat to be launched by NASA SLS, which will navigate to L2 to image the Earth's plasmasphere and observe lunar flashlights (Fig. 1). The development of the flight model is now complete and, accordingly, lessons learned have been collected for each subsystem and from the point of view of the system.

2.3. Requirements Based on Lessons Learned

The items collected in the lessons-learned survey were aggregated in a way that eliminated duplication. In summary, related to improving efficiency and enhancing the reliability of satellite design and development, 51 lessons learned are extracted. The phases of the design and development process that are strongly related to each item were identified. In addition, the contents of the items were classified into several categories to make the point of the lessons learned easily understood. Finally, this study added the functions that would be necessary to realize the summarized lessons learned. The results are shown in Table 1.

3. Concept of Operation

The concept of the tool, which fulfills the requirements mentioned in Chapter 2, is illustrated in Fig. 2. The diagram describes the functions of the satellite design and development phases and how they interact with each other using the tool. The right side of the diagram is mainly the design phase and the left side is the satellite integration and testing phase. The concept of each tool is described below.

Table 1. Category of function to realize lessons learned

Category	Function
Knowledge transfer	In the early stages of design and development, based on the expected future development, manufacturing and operation and the experience of past projects, it will provide us with the knowledge and knowledge of what issues to pay attention to in design and development (defects) and the trade-offs related to them.
Communication	Makes it immediately clear what the stakeholders are thinking and what needs to be negotiated, making it possible to visualize how much margin is allocated to which design decision and which requirements are tied to it. Reveal where specifications are interfering with each other unnoticed.
Design space exploration	Enables fast and exhaustive exploration of design trade space.
Data management	Satellite design information is version-controlled. The latest information should be available. The formatting of tables, text, and logs should be consistent and sufficient so that they can be reused when needed.
SW and Document writing	When one part of the specification is changed, the SW and text associated with it will be changed automatically. Automation of English translation and generation of simulation setting files.
Test and analysis	Regardless of whether it's hardware or software, if you define tests and the version control branch of design you want to test, it will automatically execute necessary tests and analyses to summarize the results.
Task management	Visualize who is spending how much time on what tasks, and prioritize the tasks. Make sure that the results of the tasks can be referred to later.
Verification reliability	Define the tests in advance, including hardware and SW, and ensure that they are performed accordingly.

3.1. Design Support Tool

It is a tool to implement the facilitation of communication between stakeholders during the design of a satellite and to find a higher performance and more reliable design solution. To achieve this, the following functions will be realized

- To use past knowledge and data from the database
- To be able to check the validity of the design based on the simulation
- To be able to share design changes and trade-offs among stakeholders in real-time, enabling Concurrent Engineering
- To automatically generate documentation from design information

3.2. Data Management tool

For reliable satellite development, it is necessary to manage changes in design and configuration so that the root causes of failures during development or operation can be tracked and improved. For this purpose, data changes are recorded, and the

Scenario-1										SAW										Body									
Altitude		42000	m																										
Orbit Period		5550	[s]																										
Beta Angle		3	[°]																										
Eclipse Angle		35	[°]																										
Eclipse		39	[%]																										
Initial Time beginning Sunshine		0	[s]																										
Pass Average Time		400	[sec]																										
Number of Passes		7	[]																										

Fig. 3. User interface of the operation scenario tool.

concurrent design process is called “Concurrent Engineering”. Foundry Furnace by Team Xc of NASA Jet Propulsion Laboratory is one example of Concurrent Engineering. In this study, we used tools that have already been developed as tools for concurrent engineering on a trial basis and sorted out the merits and demerits of these tools. First of all, the outline functions have been summarized in previous studies. In this study, we used Virtual Satellite¹²⁾ and Valispace.¹³⁾

Virtual satellite is a concurrent engineering tool developed by DLR.¹²⁾ It is a Java-based desktop application that allows users to build models and define parameters for components and systems through a GUI. In addition, concepts that should be commonly defined between components, such as weight and power, have been registered. Based on them, the total weight and power can be easily calculated. As a unique feature, Virtual Satellite can manage versions of models using version control tools such as git and subversion. The version control makes it possible to easily select and switch design candidates. On the other hand, the design of other subsystems is not reflected in real-time, and it is necessary to retrieve the results of the study of other subsystems from the version control tool manually.

Valispace is a web-based concurrent engineering tool recently developed by Valispace Inc.¹³⁾ It allows the user to define the system configuration hierarchically in a web browser. In addition, parameters can be defined in components. These parameters can be defined not only as a number but also as formulas like Excel so that design changes in other systems can be reflected in real-time. In addition, the tool API is open to the public and can be used with python and other web tools. It is useful to combine Valispace with other analysis tools such as Python, Matlab. On the other hand, Valispace does not use git and subversion for version control, so it is difficult to backdate or switch the design to a different branch based on the version. The above can be summarized as follows.

As for the actual use of the system, this study has not finished verifying these tools since the design phase is currently finished in the satellite project. Therefore, the verification is needed in the future.

5.3. ICD Management Tool

In order to improve the reliability of satellites, it is necessary to document the interface conditions between subsystems, between components, and between systems in the form of an Interface Control Document at the design phase. In addition, it is important to manage these documents and if there is a change

更新履歴	印刷設定
更新履歴	印刷設定
更新申請	
ステータス	承認済み
リンクシート名 Mode definition, Electrical Power_master	
フォーマット更新	

ICD (Interface Control Document)	
NSPO Electrical Power ICD Form	
Item	Content
Design of grounding	
Pin assignment	
Pin type	
Module 1	Safety / Storage
Module 2	Standby / Powered-on
Module 3	Operation
Module Definition	Module 4

Item	Unit	Designed Value
		Value
Voltage1	Nominal	○V
	Max	○V
	Min	○V
Voltage2	Nominal	○V
	Max	○V
	Min	○V
Input Voltage	Nominal	○V
	Max	○V
	Min	○V

Fig. 4. User interface of the Interface Control Document tool.

of parameter or condition, it is important to notify all related engineers about the affected area and ask these engineers to consider whether it should be re-verify or not. This is necessary for reliable satellite development. In the past, ICDs were managed as Excel files in the laboratory satellite development, and the versions were recorded as the file name and stored in a file storage server for management. On the other hand, it was difficult to check the changes in each version and the update notifications were left to the engineers, so in rare cases, the changes were not recognized and there were discrepancies in the understanding of the design. The use of online tools such as spreadsheets makes it possible to keep a history of changes, but on the other hand, it is not possible to automatically record why the changes were made and who approved them. In this study, Apps Script is used to build a spreadsheet with a sheet for storing master data, temporary data, and buttons for requesting and approving updates (Fig.4). Based on the user's login information, the tool enables to store who requested which updates, for what reason, and who approved them. In addition, by defining the related systems for each component and attribute in advance, the system can send notifications to Slack when an update is requested. The notification can be considered as a method to avoid the risk of users missing out and enable highly reliable design information management.

5.4. Verification Plan

In the systems engineering process of spacecraft development, requirement decomposition, organization, and reliable functional verification are indispensable to achieve the mission objective. In this study, a matrix that supports the systems engineering processes was implemented in a spreadsheet. The matrix can manage the following things.

- Correspondence between requirements and verification items that show that the requirements are satisfied
- Correspondence between verification items and the tests to implement them

The reason for separating tests and verification items is to allow multiple definitions of verification items to a test, and the name of the test can be used for scheduling or planning of cost.

In the implementation, the requirements table, the verification items of each system, and the table of test definitions are separately implemented. The correspondence can be checked

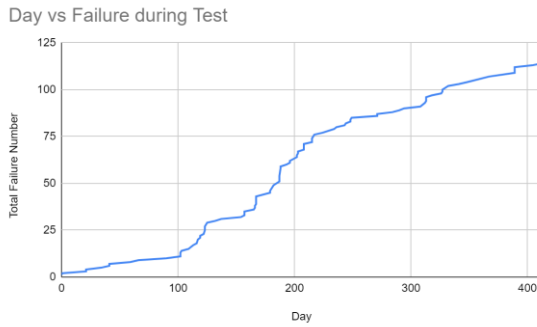


Fig. 5. Trends on the number of failures during test in ISSL 6U.

in the matrix table by using a query function for each sheet. When the detailed design of each subsystem was almost completed, it was checked whether there were any omissions in the test plan by confirming requirements and mapping them to the verification items, and by mapping the implementation of the verification items to the test plan.

In addition, the verification matrix was integrated with the test report which is described in the next section. This is because the current verification status can be displayed in the verification matrix and can be updated directly after the test results were updated. In this way, it is possible to visualize how much of the verification items have been completed in each subsystem and which subsystems are at risk in terms of scheduling.

5.5. Test Report

In this study, Google Form is used to report the test results instead of the conventional email-based reporting. This enables the accumulation of test report data in a standardized form in the spreadsheet. The form can specify the required information input so that the risk of lack of important information can be mitigated.

In addition, since the simple use of Google Form would not be able to notify the information to all related members like email, Apps Script is used to transfer the information to Slack when a form is submitted.¹⁴⁾ It enables other subsystems not attended to the test to immediately recognize the verification results, procedures, and the path of the test results. In the form, the user can enter the person who conducted the test, the test configuration, the ID of the verification item, and the result of the test (completed, retested, or aborted). In addition, when a bug occurs, it can be sent via the form as well. The report of failure also sends a separate notification to Slack, so that serious problems can be dealt with immediately.

6. Implementation Result

6.1. Usage Status

Unfortunately, the operation scenario tool is not been frequently used since the design phase of the project has been completed. On the other hand, the Spreadsheet is used to specify conditions such as attitude and sun, which is easy to operate and is being used.

As for the Design Tool, that is an experimental use of existing methods and needs to be further developed based on usability and interaction with other tools.

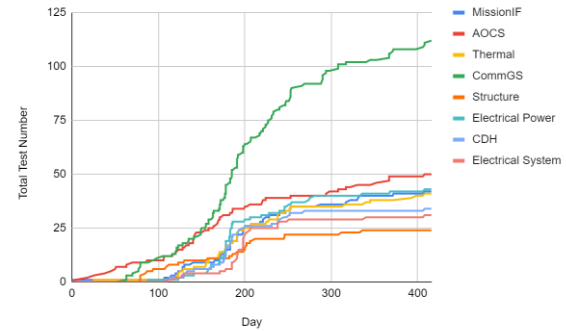


Fig. 6. Trends on the number of tests conducted by each subsystem in ISSL 6U.

As for the ICD management tool, it is hardly used except for some weight management, unfortunately. Initially, ICDs were created for each component, but now ICD sheets are being created for each field, such as weight and power. This is because the ICD sheets for each component would be huge and it would be inefficient for the management system to obtain the information. In addition, the approval system is rarely used in the ICD sheets. During the development, each subsystem implemented its original spreadsheet to manage design information and deal with the sheet as the master information. The ICD sheets are rarely updated and are merely a formality.

As for the verification plan sheet, it is used in each phase to check whether the verification items have been defined and are being executed smoothly. On the other hand, the detailed granularity of verification items is different between each system. In addition, there are many cases where updates are delayed, especially for verification items that are additionally required in the process of detailed design and testing.

Test reports are continuously used for every test. As an objective function of the tool, when and by whom the tests were conducted, the procedures and defects at that time can be managed in a unified manner. Thanks to the structured information in a spreadsheet, the result, and trends of tests can be analyzed easily. For example, the increase of failure numbers during development can be visualized such as Fig. 5. In addition, the number of test completion can be visualized (Fig. 6). It is beneficial to project managers to make decisions on human resource allocation. On the other hand, there have been cases where several subsystems are related to one test and no one sent test reports because who is responsible to send them is not decided.

6.2. Current Implementation Status

As mentioned above, this study made a plan to implement a simple spreadsheet-based tool at first and move on to the development of a full-scale web tool as a next step. Although the development of the web tool has been started, the implementation of the web tool is currently suspended due to the tight resources of satellite development itself. Web tool development requires a certain amount of learning cost, which makes it difficult to create a tool that will lead to a more efficient and reliable satellite design and development process soon. Currently, the study continues to improve and maintain the spreadsheet-based method that we have developed so far.

7. Lessons Learned

While this study has developed several tools based on spreadsheets, they are not being used effectively. In addition, this study has not been able to migrate to a full-fledged tool. This chapter discusses the causes of this situation and summarizes the lessons learned.

7.1. Stress to input data

One of the assumed reasons why the proposed method is not effectively used is that procedures such as data accumulation are time-consuming tasks for subsystem engineers. The design and verification work can be done without these tools. Therefore, even if a tool is built, if there is no direct merit to use it, it will not be used and people will neglect to enter information. In particular, students with little experience are not likely to recognize the importance of information management in the development of reliable satellites. Therefore, it is difficult for them to recognize the importance of using these tools. The benefits of these information management tools are realized only when the tools for information utilization are built. It is difficult to feel the merit in a situation where there are no functions that people cannot generally implement, such as automatic scheduling based on verification items.

7.2. Simplicity of Tool Interface is important

This study constructed a Spreadsheet-based user interface with tables. The spreadsheet has the flexibility to edit each cell in the table to change color and sort columns. On the other hand, it is not easy to restrict input to specific users, to display master data in multiple sheets by switching views, or to display hierarchical data neatly (Currently it is possible by combining Apps Script but it is not simple). In the case of validation plans and ICD approval sheets, the implementation was conducted forcibly in a spreadsheet, which resulted in poor operability and made it difficult for users to get the correct operation even if a procedure was prepared. Web tools, on the other hand, allow the user interface to be designed independently of the data, making them highly flexible. However, implementation of a full-stack web application requires implementation know-how that is different from that of satellite development, which is a heavy burden for students who mainly work on satellite development.

7.3. Counter Measures to Solve Issues

One possible countermeasure to these points is to store information in a text-based format and use git to manage it while implementing continuous testing to ensure that the information entered is error-free. Text-based input (e.g., JSON) is highly flexible and easy to input at the time of input. Based on key values in the text, it is easy to extract information and can interact with various tools more easily compared with Spreadsheet methods. On the other hand, since it cannot be highlighted or pulled down like a spreadsheet, there is a possibility of incorrect input. In response to this high error risk, software-defined tests can be prepared on a software development platform such as Gitlab to ensure that there are no errors. In order to use software development platforms, it is also necessary for users to get the know-how on the development

using Git. However, the difficulty of learning can be lower compared to the full-stack web application.

8. Conclusion

This study developed a tool to improve the efficiency of design and development for university satellite development and summarized the implementation status and lessons learned for the tool. At present, the cycle of use, feedback, and correction is not running well, and many points need to be improved in the future. In particular, the core issue is how to correct information efficiently for users (students who join in the satellite projects). The current implementation is not easy for users to see the effects of their efforts and it makes it difficult to achieve total reliability of the satellite development. To address this issue, further investigation on the use of Git, OpenMBEE, and other tools released as OSS is required.

References

- 1) Doerksen, K., and Fischer, K.: Autonomous Systems Validation (SysVal) Environment for Advancing Mission Operations., Proceeding of Small Satellite Conference, Utah, USA, SSC21-V-04, 2021.
- 2) Bryce, D., and Cappaert, J.: SmallSat Manufacturing; The Spire" Constant NPI" Model., Proceeding of Small Satellite Conference, Utah, USA, SSC19-I-01, 2019.
- 3) Austin, A., Miller, R., Murphy, J., Kolar, M., Blossom, J., Phan, T. A., and Marinan, A.: Rapid SmallSat Mission Formulation: Integrated and Concurrent Modeling in JPL's Team Xc. IEEE Aerospace Conference, Montana, USA, IEEE, pp. 1-11, 2020
- 4) Funase, R., Ikari, S., Miyoshi, K., Kawabata, Y., Nakajima, S., Nomura, S., et al.: Mission to Earth-Moon Lagrange point by a 6U CubeSat: EQUULEUS. *IEEE Aerospace and Electronic Systems Magazine*, **35**(3), (2020). Pp. 30-44.
- 5) Bayer, T.: Is MBSE helping? Measuring value on Europa Clipper, IEEE Aerospace Conference, Montana, USA, pp. 1-13, 2013.
- 6) Cameron, B., and Adsit, D. M.: Model-based systems engineering uptake in engineering practice. *IEEE Transactions on Engineering Management*, **67**, (2018), pp.152-162.
- 7) SOLIDWORKS PDM
<https://www.solidworks.com/ja/product/solidworks-pdm> (accessed December 20, 2021).
- 8) Microsoft Project
<https://www.microsoft.com/ja-jp/microsoft-365/project/project-management-software> (accessed December 20, 2021).
- 9) Google Spreadsheet
https://www.google.com/intl/ja_jp/sheets/about/ (accessed December 20, 2021).
- 10) Apps Script <https://workspace.google.co.jp/intl/ja/products/apps-script/> (accessed December 20, 2021).
- 11) Jupyter <https://jupyter.org/> (accessed December 20, 2021).
- 12) Fischer, P. M., Deshmukh, M., Maiwald, V., Quantius, D., Gomez, A. M., and Gerndt, A.: Conceptual data model: A foundation for successful concurrent engineering. *Concurrent Engineering*, **26**, (2018), pp. 55-76.
- 13) Valispace, <http://https://www.valispace.com/> (accessed December 20, 2021).
- 14) Google Form, https://www.google.com/intl/ja_jp/sheets/about/ (accessed December 20, 2021).