Proceedings of the Canadian Society for Mechanical Engineering International Congress 2020
CSME Congress 2020
June 21-24, 2020, Charlottetown, PE, Canada

Hardware and Software Project Management Best Practices for Small Satellite Systems

1st Andrada Zoltan
Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
andrada.zoltan@alumni.ubc.ca

2nd Richard Arthurs
Mechatronic Systems Engineering
Simon Fraser University
Surrey, Canada
rarthurs@sfu.ca

***Abstract*—The Command and Data Handling Team for the ORCASat CubeSat project, funded by the Canadian Space Agency, is responsible for delivering a space-ready on-board computer, supporting testing infrastructure, and ground control mission software in a three-year timeline by the launch date in 2021. Members of this team are distributed across two universities and consist of undergraduate students contributing part-time to the project. Co-lead by two individuals, the team has implemented several techniques and practices to handle the challenges that come with managing remote work. We present the methods that have been employed in the management of this team, including meeting format, team communication software, use of version control and task tracking software, and practices for long-term planning. The standardization of a design process methodology, from requirements definition to implementation, is also discussed as it has greatly helped increase the efficiency of the team as a whole.**

Many of the methods employed in the management of this team were originally based upon well-known software development methodologies, adjusted to meet the needs of combined hardware and software projects. Lessons learned from the management of previous student design team projects were also incorporated into the current management strategy. These techniques are tailored to the rigorous demands of a small spacecraft development program and have contributed to the rapid development of the project and the successes of the team thus far. Employing similar methods would be useful to any other program working under a similar timeline and team composition to that of a student-driven CubeSat development program like ORCASat.

***Index Terms*—management, design process, CubeSat**

I. Introduction

ORCASat is a 2U CubeSat project, sponsored by the Canadian Space Agency, with a three-year development timeline launching in 2021. Its mission is to provide a known-good optical reference for use in ground-based telescope calibration. The project is a collaboration between multiple universities, requiring a majority of the technical work to be completed remotely at each school. The Command and Data Handling (C&DH) team, in particular, has had members working remotely since the beginning of the project. The team is primarily composed of undergraduate students working on a voluntary basis and is managed by two co-leads, leading a team of six to seven members. The team has achieved unparalleled member retention rates, with all members currently on the team having been involved since project kickoff in 2018. As well, the team has consistently been given high praise for staying organized and meeting the milestones throughout the project thus far.

Having now hit the halfway point of the project, the C&DH team has come to recognize the techniques and management strategies that have lead to the team's success. This paper discusses these strategies, including: the design and implementation process, team management details such as meeting format and the role of the co-leads, and a discussion of lessons learned. The purpose of this paper is to present the techniques that have worked for this team in the hope that other teams can learn from and adopt them for their own benefit.

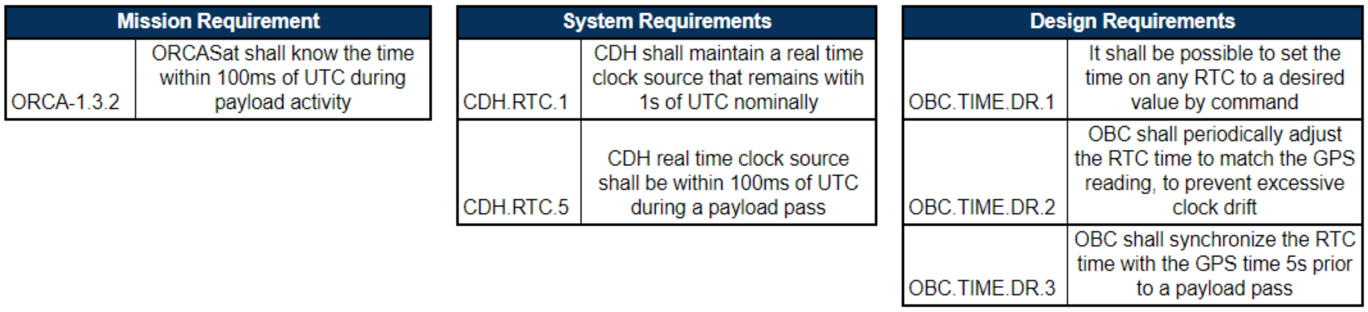
II. Design and Implementation Process

A. Concept Development

The start of any spacecraft project begins with the mission-level concept design, where the objectives of the mission are defined and the top-level requirements of the spacecraft are formulated [1]. Without this initial definition of what the final product needs to do, there is a high risk that what is actually produced does not meet the initial intent. The C&DH team has taken this philosophy and applied it down to the feature level for any hardware or software component present in the system, through the process of requirements flow down. This process involves the extension of requirements from the mission level down to the feature level, and is best performed by creative individuals that have a solid understanding of the system and its components.

Starting at the system level, mission requirements that apply to C&DH are taken and broken down into detailed system requirements. This process defines the responsibilities and capabilities of C&DH needed to make the mission succeed. During the critical design phase of the project, the same process is repeated with interfacing requirements to ensure that C&DH is developed to be compatible with the other subsystems. The step of defining system requirements should happen before beginning to prototype and design the system, as it will heavily guide the features required.

At the next level, the system requirements are taken and broken down into design requirements for each feature in the system. The purpose of these requirements is to define what the feature needs to do in order to meet the required



Mission Requirement	
ORCA-1.3.2	ORCASat shall know the time within 100ms of UTC during payload activity

System Requirements	
CDH.RTC.1	CDH shall maintain a real time clock source that remains withi 1s of UTC nominally
CDH.RTC.5	CDH real time clock source shall be within 100ms of UTC during a payload pass

Design Requirements	
OBC.TIME.DR.1	It shall be possible to set the time on any RTC to a desired value by command
OBC.TIME.DR.2	OBC shall periodically adjust the RTC time to match the GPS reading, to prevent excessive clock drift
OBC.TIME.DR.3	OBC shall synchronize the RTC time with the GPS time 5s prior to a payload pass

Figure 1. Requirements Flow Down Example

functionality of the C&DH system. Each requirement must be verifiable, as they are used to drive testing and qualification of the system later into the project. An example of the full requirements flow down for a particular feature can be seen in Figure 1.

By definition, some idea regarding the design of the feature is needed in order to write the design requirements. However, it is important to not dwell on the implementation details at this point. If implementation is defined too early in the project, it is likely to be changed as the system design matures. This will lead to wasted time and a redo of prior work. When writing design requirements, consider: how the system requirement will be satisfied by the respective technology, what constraints are imposed by the rest of the system, what level of control or autonomy the feature should have, and how the requirement may be tested.

B. Design Proposal

At this stage, requirements have been written for all features defining their role and what they must do in the system. With a clear idea of the intent of each feature, it becomes possible to create a design that meets these requirements. There is no more concern regarding what needs to be done, but rather how it should be done. This stage of the process involves developing a document that outlines the implementation details for a particular feature. In the case of a software feature, information included in these documents may be: high-level structure and components of the feature, major data structures and thread-safe primitives needed, API definition, and integration with other features in the system. Whereas the hardware design documents consist of: trade-off analysis between acceptable parts, relevant calculations, application schematic, and a literature review of the reliability of such a part.

Focus of this discussion will be placed on software design, as it spans for a longer duration of the project and requires more iterations. In a software feature design document, the details of the design are broken out over different development phases to conceptually split the implementation into manageable pieces. The phases are typically as follows:

- **Phase 1:** Core functionality of the feature is implemented, which includes the foundation needed to make the feature functional at a basic level.
- **Phase 2:** Advanced capability is added that was not initially required for a basic implementation, but is needed in the final system.
- **Phase 3 & 4:** Health check capabilities and error handling are included to allow for detection and response to error scenarios in flight.

It is naturally difficult to know all the details of how a feature will work in the different development phases from the start of the project. It is acceptable to not fully flesh out the design of the feature in phase 2 onward, but keep the description brief and high-level. These documents are meant to act as working documents and should be updated with more details at every phase as the feature matures. The goal of the documents is to outline the work that is to be done over the course of the next phase and brainstorm improvements for future phases. In general, a team member should be able to take the current phase in this document and implement it to the specification that is described.

The design phase may need to be done by leadership during the start of the project to set a standard for what is expected for these documents. Additionally, document templates should be made available to ensure a consistent level of documentation across the system. Team members who are interested in doing design work are able to do so, but may require some leadership intervention to help kick-start the work. Typically, a meeting will be held with the lead developer to clarify any questions and discuss the deliverables needed for the next phase.

Once the design has been written down, it undergoes a week of review in which team leads and other members look over the document. This early design and review process allows for fundamental flaws to be discovered without the large time investment of a full implementation. Countless hours have been saved through finding these problems early and adjusting a design accordingly. Once the design is approved, the team leads take the current phase and divide it into individual tasks that are referenced in GitLab issues. These issues serve as the unit of work when assigning to team members and should roughly span one to two weeks of work, as it ensures tasks do not draw on for too long, giving members a sense of progress and accomplishment for completing them.

C. *Implementation and Review*

With tasks documented in GitLab issues, team members are able to take an individual task and begin working on the code implementation for the respective feature. Programming is typically executed as an independent task, but members are recommended to reach out frequently for help regarding implementation details and debugging assistance.

Prior to submitting their work for code review, the implementer must complete a series of tasks preparing their implementation to be merged back to the master branch. These tasks include: re-basing the working branch with master, ensuring the code compiles on all platforms without warnings, and passing a series of automated tests that verify software functionality. These steps, along with the review process, are aimed at minimizing the issues introduced in the main code base, and keep the master branch functional at all times. Once these items are complete, members must submit a merge request and the code review process is started.

The code review is typically handled by one reviewer, but may be looked at by multiple people if the changes are significant. Reviewers are cycled through to ensure an even distribution of work among members, and provide everyone with the experience to give and receive feedback. The code review process is intended to be thorough and impose a high level of quality to all code going into the master branch. The items that reviewers look for are:

- **Coding Standards:** A selection of MISRA-C rules and custom formatting rules are used to enforce consistency in the code base [2]. The MISRA-C rules are also verified by the compiler tool-chain, which generates warnings for non-compliant code. Our coding standard includes coding style, or format rules designed to produce a readable code base and prevent certain errors. Adoption of a consistent coding style is recommended when working on mission-critical software, as it can help make certain errors more easily visible [3].
- **Efficiency and Correctness:** Are there any improvements that can be made to make the implementation more efficient? Are there any missed corner cases in the logic?
- **Ease of Use:** Is the feature easy to integrate with the rest of the system (i.e. is the API suitable)? Are there commands to control and enable this feature? How testable is this feature?

Although the code review process may seem lengthy and exhaustive, it is very critical to the development process [4]. Peer reviews can catch coding mistakes and major flaws in the design [3], expose multiple members to how a particular feature works, and enforce consistency across the entire code base. The code review process also provides a great learning experience for both the reviewers and the implementer, preparing team members for work in industry.

III. Team Management

A. *Software and Tools*

A standard set of software tools has been adopted by the team for managing development. Many are also used project-wide at ORCASat, but when the C&DH team has a choice, low-cost and cross-platform tools are preferred, allowing them to be used without the need for university networks or licenses.

- **GitLab:** GitLab is used to host git repositories for all software and hardware projects. Unlike other teams, the ORCASat C&DH team uses version control on all software projects, starting at the beginning of development [5]. Our software development operates on a continuous cycle - there are no experimental or side projects, all development is integrated in the main code base, even if the code may not have its phase 3 features implemented yet. The focus within the design process on producing features that will integrate well with the existing system makes this possible.
- **GitLab Issues:** Software bugs and development tasks are tracked in the appropriate repository using GitLab Issues. Many issues are grouped into "milestones," which represent a phase of development of a feature, or a catch-all grouping for miscellaneous issues, such as "bugs" or "nice to have." Milestones provide at-a-glance snapshots of progress on a particular feature.
  Many issues are created from the tasks identified during the in-document design of a feature. Others arise as a result of bugs, or from ideas that may need to be investigated further. Issue detail varies, but before assignment to a team member, the issue is checked and if necessary, updated with relevant details. Tracking tasks using issues that are close to the code base and can easily reference other issues, merge requests, or code, enables the code and the task list to stay synchronized. When working on multiple software projects at once, the ability to link the work and the task tracking becomes very important, and is something that cannot be done easily with more generic task management or kanban board tools.
- **Wiki:** GitLab Wikis associated with each repository are used for static information such as coding standards, how-to guides, and procedures. Since edits to wikis are not visible live, they have proven to be difficult to use for documentation that may change a lot during its creation, such as design documentation for software features. Additionally, the lack of support for comment-based reviews of wiki content means wikis are not well-suited to our design process, where review is crucial.
- **Code Composer Studio:** The integrated development environment provided by Texas Instruments is used for programming firmware, and interactive debugging. Leveraging manufacturer-provided tools ensures that a cross-platform development environment is straightforward to set up, and removes the need to maintain a build system and its installation instructions for all platforms.

- **G Suite:** The Google Docs and Google Sheets tools from G Suite by Google are used extensively for design documentation, requirements tracking, and meeting notes. The document review comment tools are used extensively during design review, and documents are organized strictly in a directory hierarchy that is shallow, which aids in finding documents easily. Additionally, the Google Sheets API is used by custom tools to generate code from items tracked in a spreadsheet, and to synchronize the test plan definition spreadsheet with the actual test implementations in code.
- **Slack:** As with many other CubeSat programs, Slack is used for project-wide communication [5]. More specifically, the C&DH channel is used only for technical and administrative discussion. Public but on-topic discussions are encouraged as they allow all team members the opportunity to track and contribute to the discussion. Other channels exist for each subsystem, as well as one dedicated to systems engineering. Team members are encouraged to ask questions relevant for a specific subsystem in the associated channel, which helps keep a record of the conversations and allows input from multiple relevant parties.

B. Meeting Format

Online meetings are held weekly in the evenings, with the time being adjusted at the beginning of every semester to account for changing school schedules. During the meeting, a set of meeting notes is updated while being presented on a shared screen, making it easy for participants to follow along. The notes from the previous week's meeting are used as the base for the current week, ensuring items on the agenda from previous weeks cannot get overlooked.

The weekly meeting is used to provide project-wide updates, facilitate verbal discussion of new issues, and to transfer tasks between stages in the development or implementation process.

The meeting format is as follows:

- **Management Update:** Updates from the previous week's project-wide meetings, typically attended by C&DH team leads. These updates are intended to provide project-wide context, and emphasis is placed on how any updates may affect the development of the C&DH system.
- **Review of Timeline:** The C&DH four-month timeline is consulted and updated to reflect changes to the plan, and to mark completion of key tasks. At the end of every month, a retrospective is also held to gain feedback from team members about efficiency improvements.
- **Per-Person Task Update:** A table is maintained with the ongoing tasks that each team member is working on. For each task and person, the status of work from the past week is updated, and a plan is agreed upon for what will be attempted during the next week. At this stage, tasks often move from the individual implementation stage into the code review stage.
- **Assignment of New Tasks:** Once the status of the current tasks has been assessed, the process of assigning new tasks begins. Code reviews are first assigned to anyone who is interested in reviewing the new work. Team members are always encouraged to volunteer to take on reviews if their interests and time allow. Otherwise, a review queue is maintained, and members are assigned to complete reviews based on their position in the queue. Overall, flexibility is maintained; if members have a busy week coming up, or would prefer instead to act as a secondary reviewer, accommodations are made willingly by team leads. Tasks are often presented to members with a focus on what could be learned by reviewing the work, and pairs of reviewers may be assigned if the area of development is particularly complex or new to one of the reviewers.

C. Team Leadership

The ORCASat C&DH team is lead by two co-leads with equal responsibility, accountability, and technical ability. This structure allows for completion of all the required management tasks while remaining flexible to availability from week to week. It also provides a level of peer review for all decisions, not just technically but in management as well. The role of the two co-leads is to drive the design of the system, delegate tasks to members, and define best practices. The co-leads define how the team is run and the process by which work gets completed.

The management strategy employed is very hands-on, in which the co-leads are also heavily involved in day-to-day programming, design work, and testing. In this way the co-leads are able to acquire a strong understanding of the system and how to develop for it, which allows team members to benefit from this added expertise. The co-leads often provide guidance to members where it is needed, and it is the leads' job to kick-start tasks so they may easily be picked up by any member. The leads are also involved in reviewing the work that members have done and providing advice for ways to improve it. Leadership divided among two people has been a very rewarding process, as working collaboratively allows the leads to learn from each other and ensures maximum knowledge transfer. This technique is one employed by many student teams, and can be highly productive with frequent communication [5].

D. Timeline Planning

Long-term planning is required to ensure the team is working on the most critical tasks and subsystem deliverables are completed on time. Planning is executed on a four-month basis, as it aligns with school semesters, and it is driven by: required deliverables, needs of other subsystems, and dependencies between features. Towards the later stages of the project, task planning may also be driven by testing requirements and the infrastructure needed to execute a pre-defined test plan.

Features should be developed as they are needed to the point that they are useful elsewhere in the system. It is not necessary, and it is possibly detrimental, to develop features beyond what is needed at the time. There are always higher priority features that require the development time instead. As well, over-developing features often leads to wasted work that becomes obsolete by the time the feature is needed.

When assessing the priorities of different tasks, there are three important factors to consider. The first factor, which is of particular concern to the C&DH system, is the development level and timeline of other subsystems. Due to the close integration of C&DH with the rest of the satellite, timeline delays in the development of other subsystems may negatively impact the C&DH team's timeline. Expectations for other subsystems should be laid out well in advance to mitigate this risk. Secondly, dependencies between features and the importance of each item must be considered. Working on dependent features in parallel can lead to large delays when trying to coordinate the work of different people. An important note is that tasks that are closely related to hardware take much longer than purely software tasks. And thirdly, it is a good idea to account for extra time in the plan for software bug fixes and other overhead that may come up.

IV. Lessons Learned

A. Remote Work

Effective remote working requires modifications to a typical engineering development cycle, and considerations in the development of team culture. The C&DH design process relies on well-supported software tools, and emphasizes unfettered hardware access for all developers. These practices allow troubleshooting to occur quickly, reducing friction within the development cycle. In terms of communication management, openness and flexibility are the key traits.

Hardware

The requirement to use custom-designed hardware has not presented a challenge to C&DH team work. All team members are given either a copy of the custom OBC hardware, or a low-cost development board with the same micro-controller used on the OBC, which can run similar firmware. The firmware for the development board implements simulated versions of hardware features that are only present on the complete OBC. From a user interface perspective, when sending commands, these boards perform identically. Since every team member has hardware, every team member can implement, debug, and test firmware features independently. This allows development work to be parallelized, and affords team members flexibility in when and where they complete development tasks.

Copies of custom hardware are allocated to team members with better access to lab equipment, or to team members working on firmware features that require custom hardware, such as low-level drivers. The custom hardware prototypes are designed to operate without lab equipment, ensuring work can be done without a well-equipped electronics lab. For example, all power supplies required during day-to-day firmware development are derived from power provided over the USB port. This USB port also provides a debug serial connection to the OBC. Lab equipment is sometimes required for specific tests, in which case a board will be tested in a university lab. However, especially in the earlier phases of firmware development, minimal lab equipment is needed relative to development time. Where lab equipment is required, the team also prefers portable and low-cost tools, such as small logic analyzers for debugging communication interfaces, rather than oscilloscopes. These techniques and preferences enable the software development, which has been recognized as a previously underestimated time sink on other projects [6], to be completed without the use of many special tools. This allows it to be completed anywhere, increasing productivity when most development is remote.

Communication

With remote work being completed by members at different times of day, most communication becomes asynchronous. Effective asynchronous communication requires consistent monitoring from all who are able to answer questions. Frequently encouraging all team members to post and answer questions in the team-wide Slack channel is an effective technique to ensure that questions are answered promptly. The importance of allowing developers to ask questions openly, and to receive responses, is well-understood in the context of educational CubeSat development programs [7]. This communication, being public to the team, is also available as a reference for anyone who needs it in the future. This is another important benefit of public communication. There is also very little non-technical or non-administrative communication in the channel. This feature of the C&DH channel increases the value of everything in it; if members see a notification from the channel, they can be quite sure that the discussion is important.

Effective remote work also requires flexibility in scheduling synchronous communication. Scheduling ad-hoc video chats is encouraged and is done frequently. These meetings are often held between team leads and a team member, and rarely include more than three people, though the entire team is always notified and invited. The agenda is flexible, but often consists of design or document review, co-troubleshooting, or review of a list of questions or pain points. Meetings of this format usually last less than an hour, but are an extremely effective way to get tasks progressing. Team members are often grateful for the opportunity for a focused discussion after they have been stuck on a problem. The key to getting this form of meeting to become frequent has been to encourage team members to consider it as an option. During task review at weekly meetings, a short, focused discussion is often an option suggested as a next step.

B. Team Retention

Team member retention is a problem that many CubeSat programs struggle with, especially those who incorporate CubeSat development into coursework [5] [6] [7]. Often, the use of professors or graduate students to provide multi-year continuity is used [5]. Strict documentation practices are also cited as a knowledge transfer tool for students coming in and out of the project [7], [5]. However, little effort seems to be placed into promoting retention of current members.

Team Motivation

Motivation, particularly in a team setting, can be a difficult thing to achieve. Every person is motivated in a different way, so using a generic method to promote team motivation may not always be successful. The first step in motivating members is discovering what people are interested in working on and finding opportunities for them to do so. C&DH members are not given tasks that they are uninterested in working on, which leads to faster completion time and higher quality of work. People also like to work with different levels of independence. Some people enjoy the challenge of designing a new feature and seeing it through to completion, whereas others would prefer to follow instructions for implementing a task. It is important to work with members and find the right balance of independence and guidance for each one. Too much independence can be demotivating if the individual is unclear on how to approach a given problem. This issue also arises when assigned tasks are large in size and open-ended. It is difficult to make progress on a task when there are no set deliverables, leading to unmotivated team members that may leave the project over time.

Other methods for motivating a team can involve showing members that they are valued and respected, by encouraging feedback on all team process changes or design decisions. Buy-in is important to getting the team working as a cohesive unit. Additionally, it is a good idea to provide regular chances for members to give feedback. Monthly retrospectives allow everyone to voice their opinion for what is hindering and helping their performance. The team can then discuss options for solving the pain points, and implement these solutions.

Team Organization

From prior experience in past CubeSat programs, we have found that team organization is another factor that can impact member retention. Disorganized teams can cause members to become disinterested in participating, and lower team performance overall. Some attributes of a disorganized team can include lack of direction, unclear or open-ended tasks, no accountability, and no process standards. By defining and following a design process, many of these issues can be resolved. It is the role of the leads to ensure members have good direction with their tasks, by communicating with them and offering to discuss ideas when needed. Accountability is another important factor for motivating members to complete their tasks. If no one is held accountable for not completing the work they were assigned, progress can heavily diminish. One method C&DH has employed is the use of a weekly counter to keep track of how long tasks have been in progress. This counter provides a visual indicator for members to see how much time has been spent on a certain task. The goal of this method is not to be harsh when the counter grows, but to emphasize the importance of upholding a member's responsibility to the team and completing their work.

V. Conclusion

Management of a technical team requires having a handle on many different aspects of the team, from people management to process development to technical design. A good team is one which is formulated around structure and good communication, enabling members to do the work they are assigned and learn in the process. It is important to work with the members of a team and discover what motivates and enables them to succeed in their work. Good organization and structure can also make members see the importance of their contributions to the project, promoting team enthusiasm and retention.

Alongside people management, a well-defined design process helps ensure consistency, and increases the likelihood that a high-quality system will be produced. It is crucial to undergo the overhead of designing a feature prior to implementing it. This allows for discussions around the proposed implementation to form, leading to an overall better design that is compatible with the existing system. An established process also enforces good documentation and implementation practices, which results in a solid final product.

Overall, it is important to find the methods that work best for your team. Team management should be an iterative process, aimed at achieving maximum efficiency and enjoyment of all parties. The practices discussed in this paper are what have been successful for the ORCASat C&DH team, but many of the key concepts, such as structured processes, good organization, and team communication, can be applied to benefit other teams greatly.

Acknowledgment

This work is supported by the generous funding of the Canadian Space Agency (CSA), as part of the Canadian CubeSat Project. Authors are grateful for the technical guidance, resources and time the CSA has donated to this project.

References

[1] J. R. Wertz and W. J. Larson, *Space Mission Analysis and Design*, 3rd ed. Hawthorne, California: Microcosm Press, 1999, ch. 1, pp. 1–18.

[2] G. J. Holzmann, "The power of ten - rules for developing safety critical code," *Computer*, vol. 39, no. 6, pp. 95–99, 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1642624

[3] B. O'Connor, *NASA Software Safety Guidebook*, nasa-gb-8719.13 ed., NASA, 2004. [Online]. Available: https://standards.nasa.gov/standard/nasa/nasa-gb-871913

[4] C. F. Kemerer and M. C. Paulk, "The impact of design and code reviews on software quality: An empirical study based on psp data," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 534–550, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4815279

[5] L. Berthoud, M. Swartwout, J. Cutler, D. Klumpar, J. A. Larsen, and J. D. Nielsen, "University cubesat project management for success," in *Proceedings of the 33rd Annual AIAA/USU Conference on Small Satellites*. Utah State University, 2019. [Online]. Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=4362&context=smallsat

[6] L. Berthoud and M. Schenk, "How to set up a cubesat project - preliminary survey results," in *Proceedings of the 30th Annual AIAA/USU Conference on Small Satellites*. Utah State University, 2016. [Online]. Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3417&context=smallsat

[7] J. A. Larsen, J. F. D. Nielsen, and C. Zhou, "On student motivation in a problem and project-based satellite development and learning environment," in *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE, 2013, pp. 923–928. [Online]. Available: https://ieeexplore.ieee.org/document/6581346