73rd International Astronautical Congress (IAC), Paris, France, 18-22 September 2022.

IAC-20-D.1.5.9

Testing of small satellite operation and software with target hardware and rehearsals

Dennis D. Langer[a*], **Sivert Bakken**[b], **Roger Birkeland**[c]

[a]*Department of Marine Technology, Norwegian University of Science and Technology (NTNU), Otto Nielsens veg 10, 7491 Trondheim, Trøndelag, Norway, dennis.d.langer@ntnu.no*
[b]*SINTEF, Brattørkaia 17C, 7010 Trondheim, Trøndelag, Norway, sivert.bakken@sintef.no*
[c]*Department of electronic systems, Norwegian University of Science and Technology (NTNU), O.S. Bragstads plass 2b, 7491 Trondheim, Trøndelag, Norway, roger.birkeland@ntnu.no*
[*]Corresponding Author

Abstract

In this paper we discuss testing of small satellite operations and software, employing target hardware and rehearsals and presenting lessons learned during the development of the HYPSO-1 (HYPerspetral Smallsat for Ocean observation) satellite. Satellites following the standard CubeSat form factor make it easy to launch and fly novel payloads for targeted applications by procuring already flight-proven subsystems or complete satellite buses. Such novel payload systems often require custom onboard software development and implementation that provide more functionality and operational flexibility, often made from scratch. However, this imposes extensive software testing requirements before launch. Software testing and rehearsal strategies, including the challenges encountered throughout developing the 6U HYPSO-1 CubeSat and the advantages and lessons learned of rehearsals, code reviews, and manual and automated testing are evaluated.

An important principle in testing is to determine if tests will cost more to implement and perform than what they will unveil to be worth. Automation in software development can help freeing time but is not useful if the time-benefit tradeoff does not work out. Hardware-in-the-loop testing ('FlatSats') should use components as close to flight hardware as possible, as opposed to development kits and possible previous versions of hardware. However this might be expensive since more copies of the flight hardware must be procured. Team members that are working on different tasks in the project have varying knowledge of the different system aspects and they have different methods of solving problems. Code reviews and rehearsing "real" operational scenarios are effective ways to learn from each other and consolidate unequal system knowledge and problem-solving methodologies. Especially of note are scenarios simulating a need to identify and fix an unexpected issue with the satellite, which was particularly engaging and forced members to think in different ways. Software updates are a planned part of the operation of the HYPSO-1 and manual testing on target hardware as part of code reviews, can be combined with rehearsing operational scenarios.
Keywords: Cubesat, Smallsat, Software Development, Rehearsals, Lessons Learned.

Acronyms/Abbreviations

Hyperspectral Smallsat for Ocean Observation 1 (HYPSO-1)
Norwegian University of Science and Technology (NTNU)
Commercial Off-the-Shelf (COTS)

1. Introduction

<ADD PREVIOUS/RELATED WORK & REFERENCES>

Previous related work on testing of operational scenarios for university CubeSats is sparse. Few published articles discuss operator training in small satellite missions.

Standardization of SmallSat form factors via the CubSat standard [1] was a necessary step towards today's COTS (Commercial Off-The-Shelf) CubeSat kits and components. COTS components are cheaper to buy compared to developing comparable systems in-house from scratch. As a consequence, acquisition of multiple units of the CubeSat components may become feasible for low budget projects.

The Small-Satellite Laboratory ("SmallSat-Lab") at NTNU developed a hyperspectral camera payload for the HYPSO-1 CubeSat, which was launched on January the 13^{th} 2022, and the satellite and payload is currently operational. The hyperspectral camera payload consists of hardware components such as frames and fixtures, optical components such as lenses, and electrical components such as power regulation and distribution, data storage and processing systems. In addition to the hardware components, extensive custom software running on the processing systems for command and control of the payload was required and thus developed. In this article we focus on the software development.

COTS CubeSat kits and components enable the setup of multiple *flatsats*. Flatsat in this paper means functional

replica of the satellite for software testing, which is set up flat on a table as opposed to stacked in the satellite frame, see Figure 1

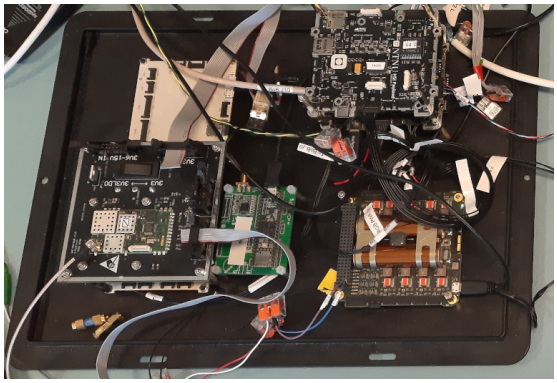

Fig. 1. Image of one of the flatsats at NTNU. Constant development and reconnections of subsystems and components results in a messy look.

FlatSat setups that are running continuously can be used for automated testing using continuous integration or other automation techniques. These can be used for component or system level regression testing, meaning to check if previous functionality is not broken by a new version of the software.

The HYPSO-1 satellite was designed to be operated by the team at NTNU using a co-located ground station and mission operations centre. Central parts of the software has been developed by students who graduated before launch. This means the current team does not have full overview of all parts of the software, and thus need training to become skilled satellite operators. The training was done through *rehearsals*. Rehearsal in this context means a planned walk-through of satellite operating procedures using FlatSat models of the satellite hardware.

The contribution of this paper is to provide insight into efficient and time-effective software development methods for custom payloads for small satellites and more efficient training methods for satellite operators.

Section 2 discusses the development, testing and training methodologies used. Section 3 presents and discusses the lessons learned. Section 4 gives concluding remarks.

2. Materials and Methods

Software development is a rapid iterative process. Feedback is received almost instantaneously as opposed to hardware development where there is can be delay from design change to finished product. This encourages experimentation in methodology and teams may converge on an optimal method in some way. A SCRUM methodology using tools from Github [2] were also used. While every developer should test their own changes to the code base, the github flow [3] methodology was used to facilitate independent testing and code reviews from other members of the team.

Early in the project, development kits with the same processing platform as planned for the flight model were available at the university and were used for initial development and testing. Development consisted of the payload processing system's firmware (bootloader, operating system, packages) and on-board processing software (drivers, telecommands). The build system of the on-board processing software for the payload was designed for both compilation and cross compilation for testing on personal computers and target hardware respectively. Their architectures are x86_64 for personal computers and armv7 for the target hardware. The processing platform also features programmable logic with which accelerator algorithms are designed. As the project progressed, multiple FlatSat replicas of the HYPSO-1 satellite were constructed to develop and test on, which replaced development kits and other approximations that have not the exact same components or interfaces as the flight model. Programming interfaces like JTAG and other access interfaces like secure shell were also removed, as those will not be available on the finished satellite.

Continuous integration techniques were used for building, unit testing and regression testing. The continuous integration framework used was Jenkins [4].

The rehearsals carried out during the project could be classified into two types. The first type is a walk-through of standard operating procedures of the satellite, to gain familiarity with the system. During development, these procedures may change in some details, but the rehearsals are nonetheless performed with the current operational model in mind. For HYPSO-1, the standard operating procedures included tasks such as taking an image, transferring files like telemetry, or performing a software upgrade, or a combination of multiple procedures. The second rehearsal type is a session where one of the team members designed an issue the FlatSat, for the other team members to troubleshoot and repair. The designed issues were, for example, no more storage space available, or corruption of various system files.

Both rehearsal types had simulated contact restrictions with the FlatSat, similar to how contact with an orbiting satellite is limited to the time while it is passing over a ground station. A low rate of random packet drops was included to simulate unreliable link conditions.

3. Results and Discussion

See Figure 2 for a high level description of how software development for HYPSO-1 was done. Rehearsals and other kinds of function testing, while keeping the system requirements in mind revealed missing features and bugs in the software, and changes in how the satellite is

expected to be operated (the *Operation model*). Tests and rehearsals were designed to verify the operation model, and the functionality of the software.

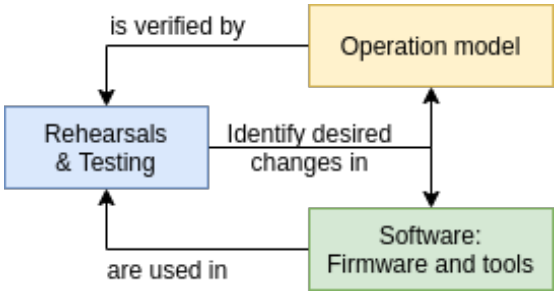

Fig. 2. Diagram of the software development process the authors converged on. Software and the operation model is tested by rehearsals and other testing procedures, through which new desired changes are identified. These can be removal of software bugs or addition of new features.

Adding a new feature to software required code review and system level functional testing and a short report by another team member via the pull request tools that Github provides. Enforcing at least one review before the feature is added, increased the discovery of bug and the quality of the code, although it slowed down development speed. Reviews sparked numerous small discussions on details of the software.

As the final target hardware design was completed and set up for use in the FlatSats, extra work was required porting the initially developed firmware to it. Even if the development kits featured the same processing platforms, they had different configurations when it came to Input/Output and memory size. This had to be taken into account on a (too) late stage in the process, causing time-consuming porting. From then on, target hardware in FlatSats was used for both development and testing.

Due to resource availability, software developers tended to fill the operator role initially, as they are most familiar with the system. Closing in on the launch, a dedicated operator sub-team was set up, as they learned the systems and spent more time operating the satellite. The software developers then could focus on other tasks.

As discussed in [5] The automated testing using Jenkins was cumbersome and time consuming to set up and maintain. The insights we got out of the automated testing were minimal and not many bugs were discovered. Then an update broke the automated testing setup, and no one in the team found it worth the time to look into the issue.

The bad results from automated testing were likely due to two reasons. The choice of jenkins as continuous integration framework and the lack of previous experience of the team members with this software and with the Java based scripting language it uses.

We found that it is recommendable to identify what work (development, testing) can be done independent on the hardware platform, in case target hardware is not available yet, or is expected to change.

In the second type of rehearsals, where an issue is introduced to the flatsat to be troubleshooted and solved by other team members do not need to concern issues that are realistic to happen on their own. They nonetheless helped the team to share knowledge, learn how the software system works, learn how to operate the satellite, find bugs and discover new desired features.

4. Conclusions

Payload development centered around FlatSats with target hardware proved valuable for the HYPSO-1 team. There was always a change in firmware needed to be done when moving from development kits to the true target hardware. Developing for target hardware to begin with, eliminates this integration step and saves time.

We would urge other projects to *gamify* their rehearsals in similar unique ways, as it boosted memorability and engagement and thus knowledge retention and transfer between team members.

Acknowledgements

This work was supported by the Research Council of Norway (RCN) through the MASSIVE project, grant number 270959, by the center of excellence Centre of Autonomous Marine Operations and Systems (NTNU AMOS) grant number 223254, and ELO-Hyp (Norway Grants contract 24/2020).

References

[1] the Cube Sat Program. *CubeSat Design Specification*. https://www.cubesat.org/cubesatinfo. (Accessed on 08/31/2022).

[2] *Github.com | service for software development and version control*. https://github.com/. (Accessed on 08/31/2022).

[3] *Github Flow*. https://docs.github.com/en/get-started/quickstart/github-flow. (Accessed on 08/31/2022).

[4] *Jenkins | open source automation server*. https://www.jenkins.io. (Accessed on 08/31/2022).

[5] Sivert Bakken et al. "Testing of Software-Intensive Hyperspectral Imaging Payload for the HYPSO-1 CubeSat". In: *2022 IEEE/SICE International Symposium on System Integration (SII)*. Jan. 2022, pp. 258–264. DOI: 10.1109/SII52469.2022.9708802.