SOFTWARE-DRIVEN APPROACH FOR IN-ORBIT DEMONSTRATION SERVICES ON BOARD OF CUBESATS

Daria Stepanova[(1)], Diego Garcia[(1)], Ravneet Kaur[(1)], Smit Patel[(1)]

[(1)] *German Orbital Systems GmbH, 10-11, eing S. Reuchlinstrasse, Berlin, Germany, 13349, info@orbitalsystems.de*

Nowadays CubeSats offer a novel, rapid and cost-effective approach to IOV/IOD missions as the solutions are heavily based on COTS materials and systems. Due to the CubeSat form factor, frequent launch opportunities are available, allowing for a significant reduction of the time needed to prepare CubeSat-based missions. CubeSat platforms decrease the cost of IOV/IOD missions by orders of magnitude as well as significantly decrease turn-around times compared to classic missions which take significantly more resources to develop.

The current state-of-the-art of IOV/IOD services struggles to balance interface definition and the growing complexity and robustness of science and commercial payload requirements, especially regarding data storage and processing. Mitigating this shortcoming takes efforts away from the quality-of-service elements: customer-friendly data acquisition, distribution processes throughout the mission lifetime, and environmental data acquisition. Lastly, the state-of-the-art has heavily struggled to meet customer launch deadlines due to the added interface with launch providers. Targeting these open points, German Orbital Systems (GOS) has developed a new, user-friendly way of qualifying the emerging materials, components, and subsystems.

With 11 successful satellite missions accomplished, GOS is proposing a new approach for payload data handling onboard IOD missions. To achieve this, a platform concept was improved together with the corresponding ground infrastructure based on the company's heritage project GROOVE. The concept is based on a 6U CubeSat, which can be shared between up to 20 customers. The proposed approach enables the allocation of payloads of different types and applications under the same architecture: from systems-on-chip to advanced propulsion modules. The service addresses such existing shortcomings starting from customer-friendly data acquisition to on-orbit data processing. To achieve long-term commercial viability and sustainability of the service, the capabilities of the platform were extended to address the flexible predefined payload-spacecraft interfaces (mechanical, electric, data) and higher flexibility of payload operability.

1 INTRODUCTION

German Orbital Systems' new IOD/IOV Service shall become one of the pillars of the company's long-term strategy. The motivation for the first activity carried out under ESA de-risk activity, was to ensure the best possible product-market fit at the end of the fully-fledged development and hence to reduce the risks of the payloads undertaking. The approach was to closely work with customers from the beginning on and this approach will be saved during the new development stage. Ensuring the best possible product-market fit from the very beginning was of fundamental importance. During the current project phase, the most critical component for the service development was improved to sustain the service for future missions to support the long-term GOS strategy.

2 PROJECT GOAL AND OBJECTIVE

The objectives of this project are driven by the shortcomings of the current state-of-the-art IOD/IOV services, feedback from 3U GROOVE customers, experience from previous missions as well as the results of the de-risk activity. The primary objectives of the project are:

- To design an independent Payload Handling Unit with high power capabilities and a broad set of data interfaces suitable for installation in the small satellite platform

- To manufacture and test a software-defined Payload Handling Unit for a small satellite platform to serve on the next GOS IOD missions
- To develop software solutions for flexible payload data handling and test the software package
- To demonstrate the capabilities of the developed module to successfully support the operations of several IOD/IOV payloads and deliver the associated payload data

At the end of the project, among others, the following results can be expected:

- Payload Handling Unit concept verified with a set of existing and potential customers.
- Payload Handling Unit software packages for the satellite functionality including Interface Controllers' software and advanced payload scheduler developed and tested
- Payload Handling Unit development kit manufactured and tested. This will be achieved by shipping the modules to existing and potential customers and receiving feedback on the processes for software development and general handling.
- Installation of the Payload Handling Unit on the mission in 2022 or 2023 and verification of its performance.

The resulting satellite architecture is outlined in Fig.1.

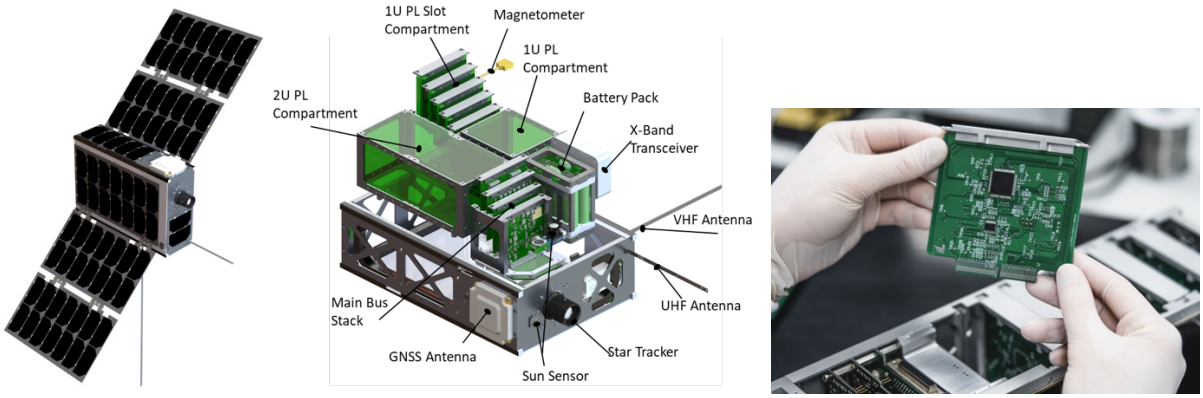

Figure 1: GROOVE EVO platform, internal architecture, and slots compartments for payloads

3 SYSTEM DESCRIPTION

3.1 Satellite component

To describe the Payload Handling Unit, it is necessary first to address the general satellite concept. The baseline satellite module consists of two main blocks: bus module and payload module. Bus module incorporates all the required instrumentation for autonomous functionality in orbit, including the Electrical Power System (EPS), Command and Data Handling Unit (CDH), Attitude and Orbit Control System (AOCS), and elements of a Thermal Control System. These subsystems oversee providing the necessary resources to the Payload Module. The Payload Module consists of payload compartments that can be organized in several different configurations and Payload Control Units (PCU) which are responsible for payload communication, control, and monitoring. A system functional diagram is depicted in Figure 2 outlining the different subsystems and the respective interface between them.

In the initial concept Payload handling unit is distributed between Bus and Payload modules.

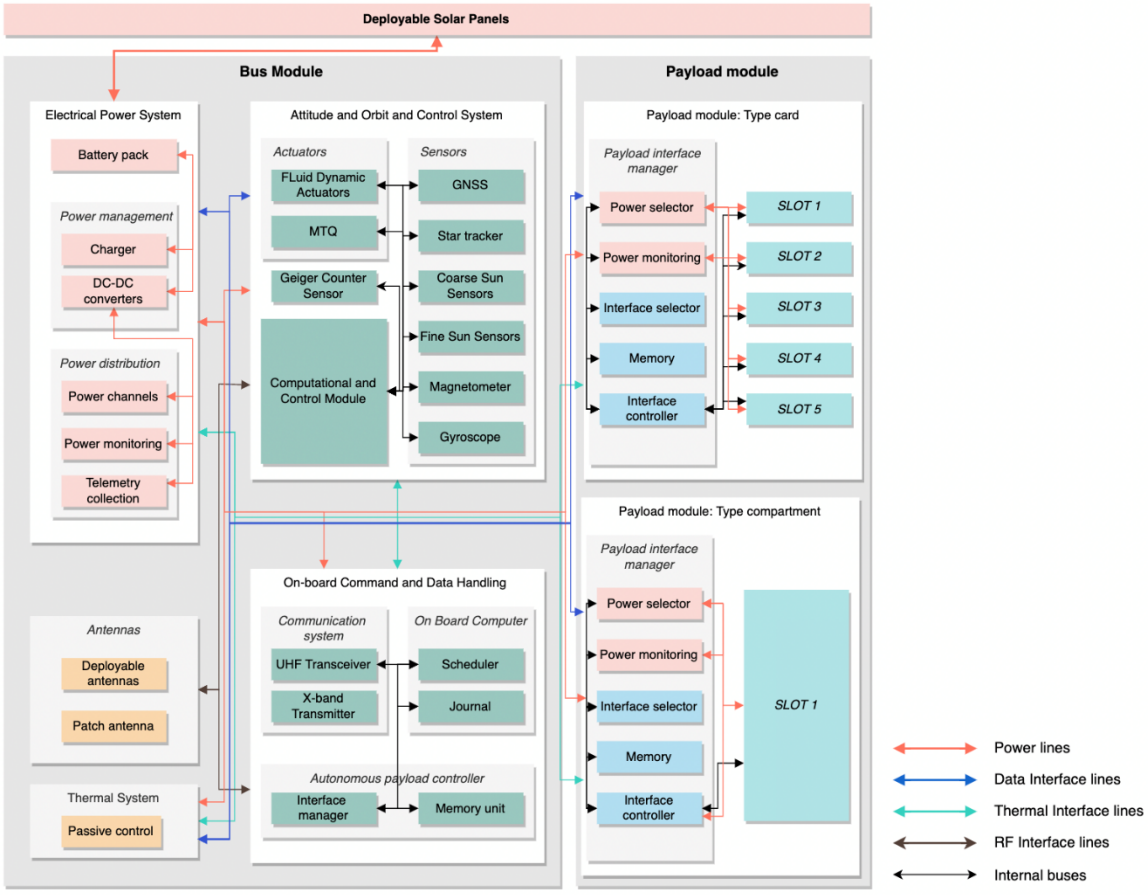

Figure 2: GROOVE EVO platform architecture

3.2 Payload Handling Unit initial concept

The Payload Handling Unit is a modular system distributed between a powerful core, several payload compartments with local power management MCUs and individual payload Interface Controllers (Figure 3). This system allows smooth and easy payload control as well as payload and main bus data exchange. Since different payloads might require vastly different approaches to data handling, a flexible data handling solution is proposed. Each payload can use its personal Interface Controller with a broad set of interfaces to access and control the payload over the satellite network.

PHU is split between the main Data Handling Unit (DHU), Compartment MCUs (CMCU), and Interface Controllers (IC). DHU is in charge of commands and data processing from and to payloads, Compartment MCUs perform power management tasks while Interface Controllers are responsible for interfacing the payloads, establishing required communication protocols as well as payload specific data handling. Payloads may also use Compartment MCUs to temporally store data or exchange it within compartments. Data stored in the compartment allows inter-payload communication if required. Should any payload require the processing of a high amount of data it can request a readout using Interface Controller.

Interface Controller will notify Data Handling Unit, and it will read data over SPI from the payload, process it, and either save it or transmit it back to the payload over an appropriate bus. All payload data from the ground is transmitted to DHU over a satellite network and stored for future use by Interface Controllers. DHU also stores all the data transmitted from payloads for long-term storage or processing and payload data queued for transmission to the ground.

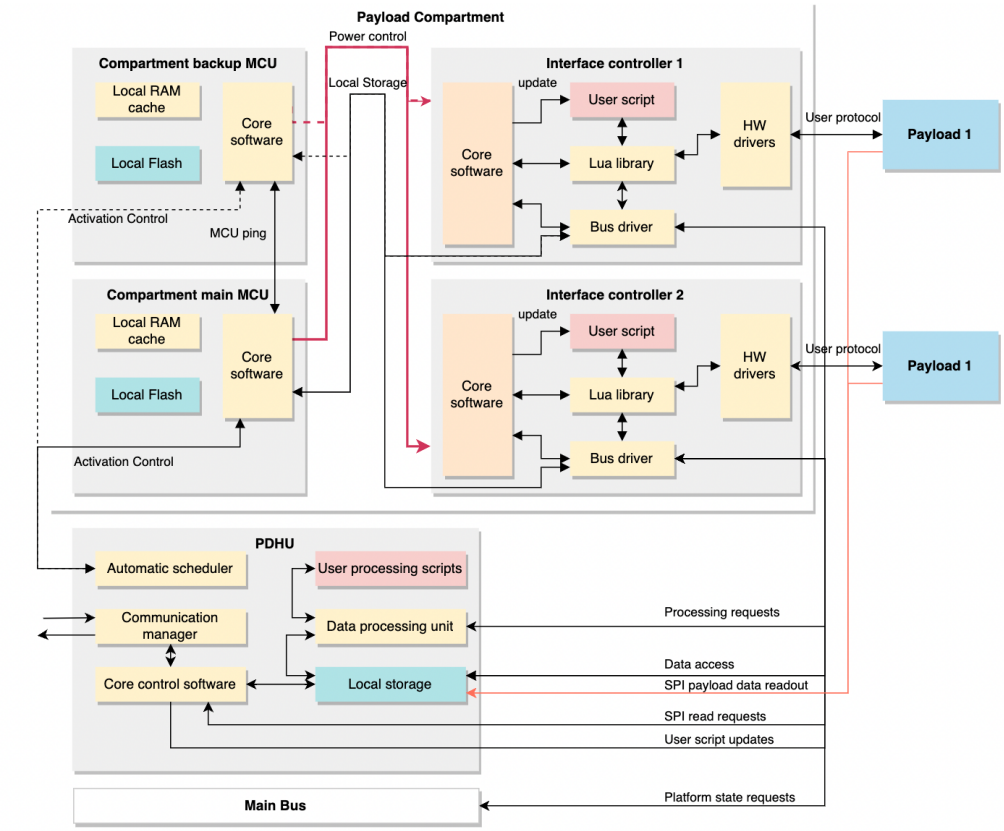

Figure 3 Payload Handling Unit architecture

3.3 Power interfaces

Different payloads might require a broad set of voltages and currents. To achieve the required level of flexibility, the power distribution core consists of a set of converters controlled via I2C by the redundant microcontrollers located on the Compartment MCUs. Digital converters shall be integrated to provide a wide range of voltages: 3.3V, 5V fixed, adjusted 3.3 – 12V, adjusted 12 – 36 V. Voltage, as well as current limits, can be programmed by the microcontrollers depending on the payload requirements. This approach of software-defined flexible power selection through adjustable converters optimizes the amount of converter required for different payloads and missions. Each payload is equipped with a power monitor and additional watchdog to mitigate possible power-related failure.

Satellite power architecture shall meet the requirements of flexibility and adjustability to accommodate payloads of different sizes and purposes. Every payload is given a set of electrical interfaces as well as several adjustable software-defined power channels. Compartment MCUs are integrated into different compartment modules to provide flexible power selection for each payload. There are 5 available buses foreseen which can be selected based on the payload requirements. Adjustable power lines can vary their voltages on the command from the Payload Power Controller Unit.

Redundant Compartment MCUs are responsible for the activation and deactivation of payloads, control power converters, and storage of payload information for inter-payload communications. They activate and deactivate payloads based on signals from PDHU and adjust power converters to provide all payloads with requested voltage levels. They also provide PHU with information about the state of all the switches and converters inside the compartment as well as monitor the power state of the payload.

3.4 Data interfaces

Each payload will have its own payload Interface Controller to establish required communication protocols as well as payload-specific data handling. Each payload compartment provides an access to the Interface Controller, which is performing the following functions:

- communication with payload
- data transfer from the satellite to the payload (time, orientation, etc)
- data transfer from the payload to the satellite memory or communication channel
- payload operational script execution
- turning sub-modules on / off
- commands execution
- telemetry collection
- access to the main bus and high-speed data downlink
- payload intermediate data storage
- power control and monitoring
- latch-up protection

The User can configure the parameters of the data interfaces according to the payload specification on different layers. For example, UART baud rate, I2C clock speed, CAN. Payload controller software implements safe access to platform network and payload developer-specific protocols and data handling; therefore, it runs both GOS and payload-specific software. This is achieved by running custom Lua scripts for each payload. Regardless of the satellite network protocol, each payload will be able to use its own protocol to communicate with the Interface Controller. The Interface Controller will provide access to the satellite network, but it will not be required for the payload developer to implement or use satellite network protocol in any form.

3.5 Payload communication

The Interface Controller is responsible for payload data handling as well as payload protocol implementation. It runs Lua scripts, which can provide access to platform information in an easy-to-use form. Such information as time, navigation, orientation, and available resources could be requested by library functions. All this information may be sent to the payload, attached to reports to the ground, or used in any other form by the user's Lua script during payload activation. The library also provides information about the payload's current state, such as power consumption and temperature.

During the duty cycle of the payload, the Interface Controller can send the data to PDHU directly or request readout over SPI using Lua functions provided in the library. Such data will be saved on the file system in a related directory and will be accessible for all future requests, and processing scripts. Should any payload require the processing of a high amount of data it could request it using Interface Controller. The Interface Controller will notify Payload Data Handling Unit, and it will read data over SPI from the payload, process it, and either save it or transmit it back to the payload over an appropriate bus.

Since scripting languages are not particularly fast in terms of performance all the computation heavy tasks are implemented inside optimized GOS C++ core software, while Lua only provides an interface to those functions. If the payload developer wants to optimize the computation-heavy part of his Lua script, it is possible to integrate any pure functions into Lua library before launch. To integrate functions into Lua library before launch payload developer should provide it in C/C++ form under a non-binding open license. It is likely that any pure functions would be easy to integrate, however it cannot be guaranteed in the case of huge, long, recurrent, or computation-heavy functions.

Instead of limiting payload developers to specific data handling solutions, GOS core software of payload controller will provide access to initialization of hardware interface, as well as access to all the data related to payload using Lua library. The payload developer will have all required data and control to provide its own data handling solution specific for the payload in Lua script. Such scripts should be provided before launch and can be easily updated afterward.

It is possible that some payload developers would prefer to adjust their own software instead of developing a custom Lua script for their payload. In such cases, the payload developer may use one of the examples of Lua scripts provided by the GOS. Custom scripts will be developed for simple and common protocols and will provide basic functionality to communicate with the ground station. GOS will also provide help to adjust such scripts or implement payload-specific scripts if the result may be added to the examples library for future use.

Library functions provide several options to implement any protocol required by the payload. These options will contain, but will not be limited to the following:

- Initialize hardware interface to PL with the selected parameters.
- Send data over a selected interface to PL.
- Receive up to the requested amount of data from the interface receive queue.
- Receive the requested amount of data from the interface with a timeout.
- Send data to be processed on PDHU over a satellite network.
- Request reading data from payload by PDHU over SPI for future processing.
- Request results of data processing from PDHU
- Temporary save data to nonvolatile memory
- Access data received from the ground during the previous communication session.
- Send data to be transmitted to the ground during the next communication session.

Providing flexible tools to create custom protocols is only part of communication controller possibilities. The payload developer may adjust Lua scripts to create complicated communication scenarios while the ability to change those scripts after launch will provide wide debug capabilities if it will be required by the payload developer.

3.6 Payload scheduling

The payload scheduler will activate the payload when its activation conditions are met, and the platform has or will have enough resources to run the full payload duty cycle, as described by the payload developer. Such activations cannot be set in any selected time method but will be performed under specified conditions. When the scheduler activates a payload, it will send a command to Compartment MCU to power up the payload and the related payload Interface Controller.

DHU stores all the updated Lua scripts for Interface Controllers, previously sent from the ground. As soon as the payload is deactivated and the updated script is available, PDHU will request the powering up of the Interface Controller in update mode without powering up the payload itself, update the script, then power down, and save the update result for telemetry.

After the deactivation of a payload by the automatic scheduler, the PDHU checks resources left from the amount requested by the payload developer for the duty cycle. This information along with other information about activation is saved in the activation report, and the report is queued for transmission to the ground. Activation reports help developers to understand the real consumption of their payloads and adjust the duty cycle description for more efficient use in the future.

4 DISCUSSION

Throughout the GROOVE EVO development project, the GOS team has evaluated technical trade-offs for the satellite design, service concept, and business model for the new small satellite-based IOD/IOV service.

The biggest risk of the activity was to develop a system for which there was little to no demand. Identifying the customers and understanding their needs was therefore a central aspect of this de-risk activity. Every detail of the new platform and of the updated service was customer-driven. Usability and user-friendliness drove the design of all service components, such as the already launched online booking platform as well as the concepts of the data analysis portal and API. It also motivated the introduction of a development kit, which allows performing checks of mechanical, electrical, and data interfaces as well as payload software development and debugging as a standard, complimentary service component.

The results of every project stage were shared with the list of potential customers and verified during interviews and questionaries. The final evaluation of the approach has shown that the satellite concept meets the IOD/IOV requirements of potential users. The set of final interviews was conducted, during which the general mission and satellite characteristics were evaluated. Customers mentioned that the platform covers the needs of the broad payload spectrum including the types of payloads they are working on. They also acknowledged that the satellite design is optimal for an IOD/IOV mission. Based on their opinion, the concept has met requirements expectations for 9.4/10. Timeline and mission schedule expectations were rated at 6.4 (1 – too slow, 5 – optimal, 10 – too fast).

All the interviewees rated the interface design higher than 8/10. Regarding the interface's improvement, most of the interviewees did not have any proposals, but the feedback about the nominal unregulated voltage bus was received and thus the additional analysis was conducted. As a result, the battery voltage level has been increased to optimize power dissipation and power distribution characteristics of the bus. Autonomous scheduler has been also evaluated high and while the scripting approach was found potentially beneficial for most payload providers, some of them could not give a solid answer. This means that the concept shall be discussed in more detail in the Interface Control Document and comprehensive examples shall be provided.

5 CONCLUSION

While the small satellite industry continues to develop rapidly, more and more systems shall be verified in near-Earth conditions to serve the mission. Developing a highly flexible and software-defined module for payloads handling is not only of interest for CubeSats but for bigger platforms too. We believe that the software-defined payload handling unit will improve the service portfolio of German Orbital Systems enabling more payloads onboard the CubeSat and more advanced in-orbit demonstration missions.

6 REFERENCES

[1] Villela, Thyrso, Cesar A. Costa, Alessandra M. Brandão, Fernando T. Bueno, and Rodrigo Leonardi. "Towards the thousandth CubeSat: A statistical overview." International Journal of Aerospace Engineering 2019 (2019).

[2] Poghosyan, Armen, and Alessandro Golkar. "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions." Progress in Aerospace Sciences 88 (2017): 59-83.

Marinan, Anne Dorothy. "From CubeSats to constellations: systems design and performance analysis." PhD diss., Massachusetts Institute of Technology, 2013.

[3] Guzman, A., S. Pliego, J. Bayer, Y. Evangelista, G. La Rosa, G. Sottile, S. Curzel et al. "The Payload Data Handling Unit (PDHU) on-board the HERMES-TP and HERMES-SP CubeSat Missions." In Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray, vol. 11444, pp. 844-854. SPIE, 2020.

[4] Schor, Dario, Jane Scowcroft, Christopher Nichols, and Witold Kinsner. "A command and data handling unit for pico-satellite missions." In 2009 Canadian Conference on Electrical and Computer Engineering, pp. 874-879. IEEE, 2009.
[5] Asundi, Sharan A., and Norman G. Fitz-Coy. "Design of command, data and telemetry handling system for a distributed computing architecture CubeSat." In 2013 IEEE Aerospace Conference, pp. 1-14. IEEE, 2013.