Materials Research Proceedings 33 (2023) 239-246 https://doi.org/10.21741/9781644902677-35

Development of a flat-sat software for deep-space autonomous GNC hardware-in-the-loop testing

Davide Perico[1,a *]

[1]Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, 20156, Milano, Italy

[a]davide.perico@polimi.it

Keywords: Autonomous Interplanetary CubeSats, Hardware-In-the-Loop, Flat-Sat On-Board Computer, Software Development Workflow

Abstract. This paper presents the development of software to integrate autonomous Guidance, Navigation, and Control algorithms and perform Hardware-In-the-Loop testing in the EXTREMA Simulation Hub facility to leverage the technology for self-driving deep-space CubeSats. Firstly, the design drivers are derived, and a multi-layered modular architecture for real-time execution is justified accordingly. Moreover, the combination of software and target computing hardware is identified by presenting the adoption of a board that satisfactorily represents the limited computational power typical of CubeSats and drawing the path towards reconfigurable computing.

Introduction

At the present time, the space sector is thriving, and in the next future an important growth in the number of space missions is forecasted [1]. CubeSats participate as one of the major actors in this trend, thanks to their concept that inferences cost-effective and relatively fast access to space. Moreover, the combination of these features with the recent advancements in on-board technology have pushed the adoption of CubeSats also in deep-space missions [2].

However, the classical paradigm for which missions strongly rely on ground-based operations could halter this course. The reliability that ground support guarantees comes at the price of high costs that usually characterize a big portion of the total budget required to design, launch, and operate a mission. Moreover, the high distances characterizing interplanetary scenarios and a forecasted growth of missions data rates ask for large ground antennas, but nowadays the NASA Deep Space Network (DSN) and the ESA Estrack stations are the only ones able to provide performant enough equipment to satisfy these requirements [3]. The result could be a saturation of the existing stations. Finally, another issue is represented by the complexity of robotic operations required for the probe: as that is increasing, the need for real-time commanding becomes paramount.

EXTREMA (Engineering Extremely Rare Events in Astrodynamics for Deep-Space Missions in Autonomy) challenges the current paradigm by enabling self-driving interplanetary spacecraft. Deep-space Guidance, Navigation, and Control (GNC) applied in a complex scenario is the core subject of EXTREMA. EXTREMA finds its foundations on three pillars. Pillar 1 regards autonomous navigation. Pillar 2 concerns autonomous guidance and control. Pillar 3 deals with autonomous ballistic capture. The project has been awarded a European Research Council (ERC) Consolidator Grant in 2019.

The product of each Pillar targets the integration in the EXTREMA Simulation Hub (ESH), a Hardware-In-the-Loop (HIL) facility that enables testing to verify and validate autonomous Guidance, Navigation and Control (GNC) algorithms and architectures [4].

The HIL simulations framework is composed of three main elements: Realistic Experimental faciliTy for vision-based NAvigation (RETINA) [5], EXTREMA THruster In the Loop

Materials Research Proceedings 33 (2023) 239-246 https://doi.org/10.21741/9781644902677-35

Experiment (ETHILE) [6], and SpacecrafT Attitude SImulation System (STASIS) [7]. Finally, SPESI oversees the process since it simulates the space dynamics and environment [8].

Moreover, all those systems interface with the EXTREMA flat-sat that reproduces the autonomous spacecraft functionalities. The flat-sat On-Board Computer (OBC) is its computing core and dwells in the STASIS air-bearing platform, which simulates the probe's attitude dynamics. The whole simulation framework is reported in Figure 1.

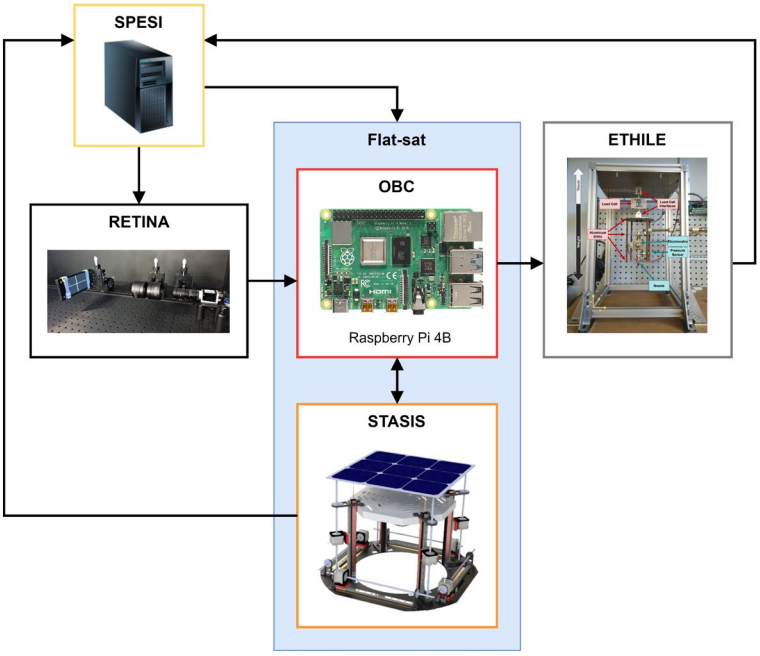

Figure 1: ESH functional breakdown structure including the flat-sat [4,5,6].

The research work focuses on the development of the OBC of the EXTREMA ESH flat-sat to enable closed-loop HIL testing for autonomous GNC algorithms. Therefore, the following research question is shaped:

Which combination of embedded software and hardware architecture is effective in managing the execution of autonomous guidance, navigation, and control algorithms in a Hardware-In-the-Loop facility?

Methodology

The approach followed to develop the flat-sat OBC software is the use of C++ as the main programming language because it offers a deeper level of optimization, and portability to different processor architectures, and it is object-oriented, which makes the code portable, reusable, and flexible.

On the other hand, a branched line of development of the software with Rust is foreseen because even if the latter language is still quite limited in libraries due to its relatively new introduction, it has interesting features for embedded programming such as direct control over running time and memory usage [9].

Furthermore, Figure 2 illustrates the workflow followed to develop the flat-sat OBC, highlighting the verification and validation sides.

The process started with the reception of the requirements from the EXTREMA Pillars and the simulation environment to translate them into the architecture design. Three main requirements were identified considering the simulation objectives of the ESH: software modularity, event-driven state change, and real-time task execution [4].

Materials Research Proceedings 33 (2023) 239-246 https://doi.org/10.21741/9781644902677-35

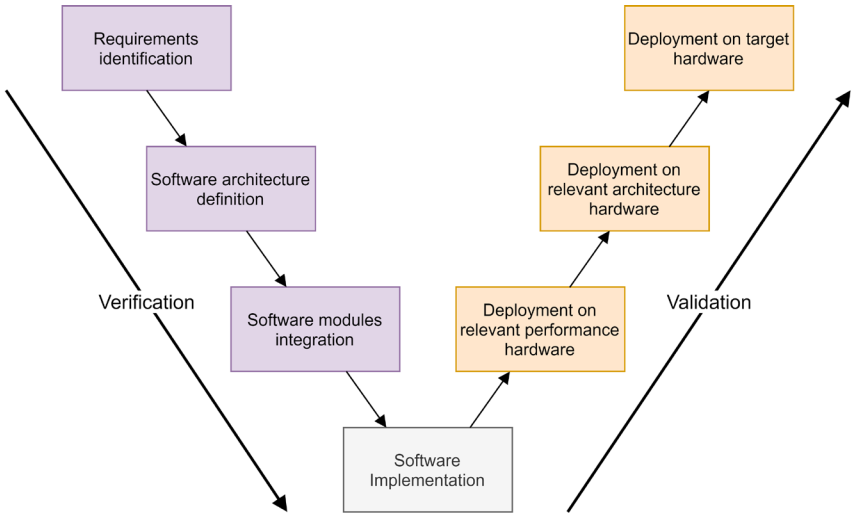

Figure 2: Flat-sat OBC development workflow.

Modularity. It guarantees an agile development of the different modules and an eased integration into the OBC. As the Pillar algorithms are designed using MATLAB and undergo continuous modifications and improvements, the MATLAB Coder is used to generate the C/C++ libraries required for their integration into the core software. The latter is based on wrappers and linking interfaces.

Event-Driven. The autonomy of the GNC algorithms has to be coordinated at the system-level. For instance, ESA definitions of execution autonomy are considered. The current trend is to target levels E3 and E4 which foresee event-based autonomous operations, execution of on-board operations control procedures, and goal-oriented mission re-planning[1]. Considering this framework, processes execution based on events moves towards these directions.

Real-Time. The simulation must happen in real-time as the software Pillar units are characterized by hard time constraints [7], therefore the solution identified was the adoption of a Real-Time Operating System (RTOS) as the operative system on top of which the flight hardware runs [10].

The second step aimed to translate the requirements into the software architecture. From literature, different architectures have been evaluated, such as MEXEC [11], FRESCO [12], Basilisk [13], and SAVOIR Onboard Software Reference Architecture (OSRA)[2].

The comparison is synthesized into the architecture reported in Figure 3, which conceives three layers: Middleware, Application and Hardware Driver.

[1] ECSS-E-ST-70-11C Space Segment Operability
[2] OSRA Onboard Software Reference Architecture

Materials Research Proceedings 33 (2023) 239-246 https://doi.org/10.21741/9781644902677-35

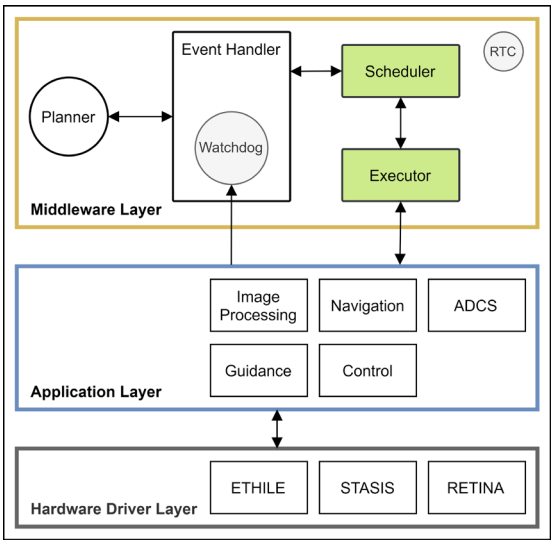

Figure 3: Flat-sat OBC software architecture.

Middleware layer. It is an abstraction layer that hosts the Planner, generating what procedures the spacecraft shall follow and holding its state. This is again beneficial for spacecraft autonomy and can be based on plans generated by a decision-making process or time-scheduled [14]. Consequentially, the plan is sent to the Event Handler, which processes the events associated with it and matches them with the corresponding tasks. Also, the Modules in the Application Layer can generate events, and the same logic is applied. The Event Handler is also responsible for generating a state transition condition to be sent to the planner. Furthermore, the scheduler receives the tasks to be executed from the Event Handler and adds them to a priority-dependent row of processes. A demonstrative scheme of a sequence of actions performed in the middleware layer is reported in Figure 4 where the change of state in the Planner is triggered by the event handled.

Materials Research Proceedings 33 (2023) 239-246 https://doi.org/10.21741/9781644902677-35

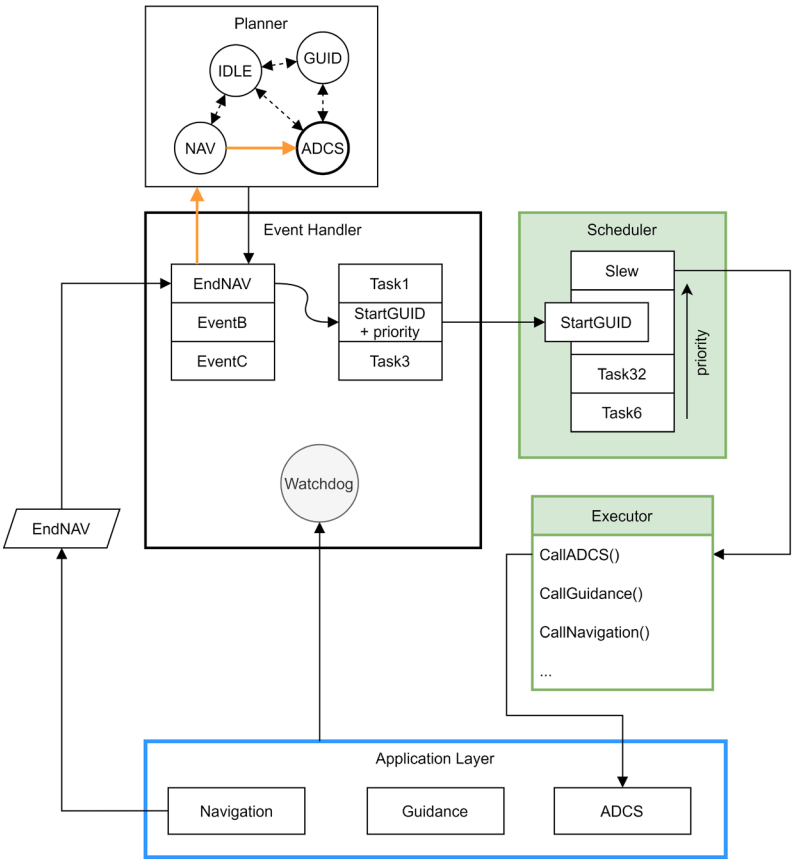

Figure 4: Demonstrative example of the functioning of the Middleware Layer.

Application layer. The application layer collects all the modules developed within the EXTREMA Pillars, as well as the ADCS algorithms. Moreover, specific manually coded wrappers and binding functions are also included in this layer.

Hardware driver layer. This last layer hosts the drivers/interfaces to communicate with the ESH hardware units to send commands and receive data such as hardware state or physical measurements if available from the sensing units.

Once the software core is prototyped, the following steps focus on the development of the remaining software modules and the deployment of the target hardware.

The process moved on to the coding of the required atomic modules of the application layer, together with the agents of the middleware layer. Object-oriented programming is widely used to define the agents and their methods. The execution policy grows as the integration of the functions goes on. The autocoded GNC modules undergo software unit tests to ensure the proper functioning and then communicate with the software middleware through wrappers that allow to eventually convert Coder's data types into standard C++ ones.

At this point the software is deployed on a target embedded hardware that has limited resources and is representative of a real on-board computer. For this purpose, a Raspberry Pi 4 Model B aided with an external Real-Time Clock (RTC) is used because it offers a relatively low-cost and a widely supported hardware. The clock is synchronized at the beginning of each simulation by communicating with SPESI such that initial time synchronization between the environment model and the spacecraft is achieved. The deployment of codes to the board is achieved in two principal ways, targeting the Linux-based operating system installed: direct compilation on the board connected through an SSH tunnel to the host computer and cross-compilation with a toolchain.

The Raspberry Pi 4B board model preliminary considered has a quad-core ARM processor clocking at 1.5GHz and 4Gb of RAM. These characteristics, as well as the power absorbed, are not typical for real CubeSat hardware which at the current state of the art has processor clock frequencies up to hundreds of MHz and can rely on low-power computing hardware [15].

Considering these limited on-board resources, having flexible and power-efficient hardware and performant software would be beneficial but hard to achieve with an independent design of the two. Hence, making the computer architecture reconfigurable, that is, rearrangeable on demand at run-time, can provide a flexible, performant, and low-power solution for autonomy because the architecture can be adapted to the algorithms that are executed [16]. An architecture of this type, promising for space applications, is System-on-a-Chip (SoC), whose reconfigurable part is often a Field Programmable Gate Array (FPGA). The different modules of the Application Layer can be optimized for execution using a different configuration of the hardware, even if paying the price of reconfigurability delay and data storage management during the process.

The current SoC under evaluation is the ZedBoard, a development board based on the Zynq-7000 architecture that is shown in Table 1, which compares different boards, some of which are used in deep-space CubeSats OBCs. The comparison highlights that the ZedBoard is a good representative of flown hardware, at relatively low cost. Moreover, the deployment of codes for reconfigurable computing relies on a High-Level-Synthesis Tool (HLS) [17].

Table 1: Relevant examples of SoC boards.

SoC	Processor	Device	Frequency	Power
Gaisler GR712RC[3]	Dual-Core LEON3-FT	Spaceteq OBC-GR712	80 MHz	5 W
Microsemi Smart Fusion-2[4]	ARM Cortex-M3	AAA Clyde Space Kryten-M3	166 MHz	2 W
Xilinx Zynq 7020[5]	Dual-Core ARM Cortex-A9	Xiphos Q7	766 MHz	2 W
Xilinx Zynq UltraScale+[6]	Quad-Core ARM Cortex-A53	Xiphos Q8	1.2 GHz	25W
Xilinx Zynq 7030[7]	Dual-Core ARM Cortex-A9	GomSpace NanoMind Z7000	800 MHz	2.3 W
Xilinx Zynq XC7Z020-1CLG484C[8]	Dual-Core ARM Cortex-A9	Avnet ZedBoard	100 MHz	N/A
Xilinx Zynq XC7Z020-CLG484-1[9]	Dual-Core ARM Cortex-A9	Xilinx ZC702 Evaluation Board	200 MHz	N/A

The final step in the design of the EXTREMA flat-sat OBC software is its deployment to a processor architecture that would normally be used on a deep-space CubeSat that would also include a reconfigurable part and more than one core. The family of processors matching these characteristics is the LEON, also adopted by ESA[10]. Specifically, a LEON3 SoC will be targeted

[3] Spaceteq OBC-GR712 datasheet
[4] AAC Còyde Space Kryten M3 datasheet
[5] Xiphos Q7 datasheet
[6] Xiphos Q8 datasheet
[7] GomSpace NnaoMind Z7000 datasheet
[8] Avnet Zedboard datasheet
[9] Xilinx ZC702 Evaluation Board datasheet
[10] ESA Onboard Computers and Data Handling Microprocessors

as it comes with debugging tools, compilation toolchains for RTEMS real-time operative system, and its LEON3-FT fault-tolerant version has already got flight heritage (e.g., on the OBC FERMI by Argotec [18]).

Conclusions

In this work, the architecture and development flow of the EXTREMA flat-sat OBC is presented. The software is used to execute extensive closed-loop Hardware-In-the-Loop simulation campaigns at the EXTREMA Simulation Hub with the objective of verifying and validate Guidance Navigation and Control algorithms for deep-space missions employing autonomous stand-alone CubeSats.

The procedure that is followed is illustrated starting from the definition of the coding language and proceeding with the explanation of the different elements composing the architecture layers. Finally, the different solutions for the target hardware are presented, as well as the novelty of the reconfigurable computing paradigm for an improvement in the implementation and execution of the on-board software.

Acknowledgments

I would like to thank my supervisor, Prof. Francesco Topputo (Politecnico di Milano), Dr. Gianmario Merisio (Politecnico di Milano), and Dr. Gianfranco Di Domenico (Politecnico di Milano) for their assistance.

This research is part of EXTREMA, a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 864697).

References

[1] A. Freeman and C. Norton. Exploring our Solar System with Cubesats and Nanosats. In: Proceedings of the 13th Reinventing Space Conference. Oxford, UK, 2018. https://doi.org/10.1007/978-3-319-32817-11

[2] E. Kulu. Nanosatellite Launch Forecasts - Track Record and Latest Prediction. In: Small Satellite Conference 2022. Logan, Utah, USA, 2022.

[3] L. J. Deutsch, S. A. Townes, P. E. Liebrecht, P. A. Vrotsos and D. M. Cornwell. Deep Space network: The Next 50 Years. In: 14th International Conference on Space Operations. Daejeon, Korea, 2016. https://doi.org/10.2514/6.2016-2373

[4] A. Morselli, G. Di Domenico, E. Andreis, A. C. Morelli, G. Merisio, V. Franzese, C. Giordano, F. Ferrari and F. Topputo. The EXTREMA Orbital Simulation Hub: a Facility for GNC Testing of Autonomous Interplanetary CubeSat. In: 4S Symposium Proceedings. Vilamoura, Portugal, 2022.

[5] P. Panicucci, E. Andreis, F. Franzese and F. Topputo. An overview of the EXTREMA deep-space optical navigation experiment. In: 3rd Space Imaging Workshop. Atlanta, Georgia, USA, 2022.

[6] A. Morselli, A. C. Morelli and F. Topputo. ETHILE: A Thruster-In-the-Loop Facility to Enable Autonomous Guidance and Control for Autonomous Interplanetary Cubesats. In: 73rd International Astronautical Congress (IAC 2022). Paris, France, 2022.

[7] G. Di Domenico and F. Topputo. STASIS: An Attitude Testbed for Hardware-in-the-Loop Simulations of Autonomous Guidance, Navigation, and Control Systems. In: 73rd International Astronautical Congress (IAC 2022). Paris, France, 2022.

[8] C. Giordano and F. Topputo. SPESI: A Real-Time Space Environment Simulator for the EXTREMA Project. In: 33rd AAS/AIAA Space Flight Mechanics Meeting. Austin, Texas, USA, 2023.

[9] N. D. Matsakis and F. S. Klock. The Rust Language. ACM SIGda Ada Letters, 34 (3) 103-104. 2014. https://doi.org/10.1145/2663171.2663188

[10] P. Hambarde, R. Varma and S. Jha. The Survey of Real Time Operating System: RTOS. In: 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies. Nagpur, India, 2014. doi: https://doi.org/10.1109/ICESC.2014.15

[11] M. Troesch, F. Mirza, K. Hughes, A. Rothstein-Dowden, A. Donner, R. Bocchino, M. Feather, B. Smith, L. Fesq, B. Barker and B. Campuzano. MEXEC: An Onboard Integrated Planning and Exécution Approach for Spacecraft Commanding. In: 30th International Conference on Automated Planning and Scheduling. Nancy, France, 2020.

[12] R. Amini, L. Fesq, R. Mackey, F. Mirza, R. Rasmussen, M. Troesch and K. Kolcio. FRESCO: A Framework for Spacecraft Systems Autonomy. In: 2021 IEEE Aerospace Conference (50100). Big Sky, Montana, USA, 2021. https://doi.org/10.1109/AERO50100.2021.9438470

[13] M. Cols Margenet, H. Schaub and S. Piggott. Flight Software Development, Migration, and Testing in Desktop and Embedded Environments. Journal of Aerospace Information Systems, 18 (4) 157-174, 2021. https://doi.org/10.2514/1.I010820

[14] E. Andreis, P. Panicucci, V. Franzese and F. Topputo. A Robust Image Processing Pipeline for Planets Line-Of-sign Extraction for Deep-Space Autonomous Cubesats Navigation. In: 44th AAS Guidance, Navigation and Control Conference. Breckenridge, Colorado, USA, 2022.

[15] A. D. George and C. M. Wilson. Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites. Proceedings of the IEEE 106(3) 458 – 470. 2018. doi: https://doi.org/10.1109/JPROC.2018.2802438

[16] Z. Wan, A. Lele, B. Yu, S. Liu, Y. Wang, V. J. Reddi and A. Raychowdhury. Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities. arXiv preprint. 2022. https://doi.org/10.48550/arXiv.2205.07149

[17] G. W. Donohoe and J. C. Lyke. Reconfigurable Computing for Space. In T. A. Thawar, editor, Aerospace Technologies Advancements, chapter 3. InTech, 2010. ISBN: 978-953-7619-96-1. https://doi.org/10.5772/117

[18] V. Marchese, N. R. Benigno, S. Simonetti, E. Fazzoletto, F. Miglioretti, V. Di Tana, S. Pirrotta, M. Amoroso, E. Dotto and V. Della Corte. LICIACube Mission: The Fastest Fly-By Ever Done by a CubeSat. In: Small Satellite Conference 21. Logan, USA, 2021.