ThalesAlenia Space
a Thales / Leonardo company

esa

→ DISCOVERY

Enhanced Observability of OBSW for enhanced testing

Executive Summary Report

Study

OSIP Campaign: NEW CONCEPTS FOR ONBOARD SOFTWARE DEVELOPMENT

Affiliation(s): Thales Alenia Space France

Activity summary:
The main objective of this activity is to propose ways of increasing the flight software observability capabilities and complementing current testing techniques thanks to enhanced mechanisms provided by modern microprocessors and operating systems. The outcomes of this activity are therefore: an state of the art of existing observability mechanisms, a list of proposals on how to use the identified observability mechanisms during testing but also during the operation of the satellite, as well as recommendations to enhanced current development, integration and testing of on-board computers.

Publishing Date: 21/03/2025
Contract Number: 4000140065/22/NL/GLC/ov
Implemented as ESA Express Procurement

→ THE EUROPEAN SPACE AGENCY

ESA Discovery & Preparation
From breakthrough ideas to mission feasibility. Discovery & Preparation is laying the groundwork for the future of space in Europe
Learn more on www.esa.int/discovery
Deliverables published on https://nebula.esa.int

1 Introduction

A major concern when dealing with test, validation and in-flight fault diagnosis of On-Board Software, whose complexity is continuously increasing, thanks notably to the increased computational power provided by modern microprocessors, is the level of observability that is provided to software and hardware engineers. In general terms, the more observability is provided to the engineering teams, the easier it is to diagnose an anomaly.

Classical observability mechanisms are limited and do not provide the needed observability of nowadays avionics systems. This activity aims at defining innovative observability mechanisms that will be able to answer nowadays observability needs.

2 Methodology

In the first phase of the activity, the observability needs associated to the different OBSW lifecycle phases, but also associated to the growing use of complex hardware and software architectures, have been analysed in order to identify the set of observability needs for which the activity shall identify observability mechanisms.

In order to answer to the identified observability needs, the activity has performed an analysis of enhanced observability mechanisms provided by modern hardware architectures, but also by software architectures considering modern RTOS and hypervisors, including Linux operating system since it is already being used in space projects around the world.

It is important to clarify that the performed analysis covers only part of the existing hardware platforms, operating systems and Linux observability tools, as this activity was not meant to thoroughly analyse all the different possibilities. However, the produced document is published in an open format so that it can be completed in future activities and projects, please contact the Flight Software Section (TEC-EDS) within ESA/ESTEC if you want to know more.

On the other hand, storage capacity has always been very limited in space due to the sensitivity to radiation of embedded memories and the mitigation mechanisms implemented on them (ECC, interleaved bits, row/column redundancy). Therefore looking at the large amount of observability data that could be produced by the mechanisms identified by this activity, it was necessary to analyse the current status on on-board data storage types and capacities as well as analysing the possible retrieval links that could be used to retrieve the stored information during the whole lifecycle of the satellite.

The analysis of the innovative observability mechanisms and the storage and retrieval techniques allowed to propose the following set of mechanisms and techniques to enhance the observability of flight and mission software during all phases of the satellite lifecycle:

- Standardisation of boot report
- Increase information of death reports
- Add logging capabilities to the execution platform
- Monitoring of system statistics
- Increase the use of PUS Parameter Statistics Reporting Service (service 4)

- Make use of existing Operating System observability mechanisms
- Increase use of auto-tests in OBSW
- Improve test means
    - Increase the use of hardware probes
    - Foresee a trace logic analyser for hardware platforms without support of hardware probes
    - Dedicate a serial communication link to dump real-time tracing of the software execution
    - Enable big data in test benches
- Enrich existing validation methods
    - Analysis of TM in real-time
    - Perform checks on logged data
    - Stress and robustness tests
    - Endurance tests
    - Dedicated test partition
    - Fault injection
    - Operational tests

On top of the proposals already presented in this document, the activity has highlighted some shortcomings in current processes through the lifecycle of a satellite and provides recommendations such as the setup of a formal integration test campaign, the adoption of HW/SW co-engineering activities, the integration of hardware probes into test means and the maintain of debug capabilities (and tracing if possible) as late as possible during the validation and verification of the satellite.

3 Conclusion

In conclusion, the comprehensive analysis of observability mechanisms provided by modern microprocessors and operating systems has highlighted their significant potential in enhancing the development, and testing phases of flight software. This activity has shown how the identified observability mechanisms can be used to enrich existing testing techniques but also to increase overall observability of the flight software while in operation.

By integrating these observability mechanisms and recommendations from the early stages of the software development lifecycle, it is possible to achieve early identification and resolution of issues, enabling a proactive approach to quality assurance. Moreover, integrating the proposed observability mechanisms in the flight software will enable a deeper understanding of the software health and performance but also provide valuable information to engineers to understand any anomaly that raises during the operation of the satellite.

To achieve these objectives, the next section will present a list of follow-on activities that will help bringing the recommendations and proposals presented in this document to industrialisation.

4 Future Work

Following the completion of the current study, several follow-on activities have been identified to further enhance and build upon the insights gained.

Activity	Comments
Share recommendations on boot and death reports with SAVOIR	Main objectives: - Define parameters to be included in both reports - Define where to store the reports and how to downlink them either as part of a standard or as a handbook
Share TN1 “Analysis of innovative observability mechanisms” in future activities and missions	- Specific implementations of the performance monitors, CoreSight system and tracing retrieving options on ARM-based processors - Consider other architectures such as PowerPC, SPARC and RISC-V - Consider other RTOS/Hypervisors such as XtratuM, FreeRTOS or VxWorks - Consider other Linux observability and monitoring tools
Analysis and evaluation of minicoredumper	To check: - Maturity of the code - License associated to the source code - Possibility to integrate it into space software architecture (not Linux) - Evaluation of the size and content of the generated reports depending on the tool configuration - Footprint of the tool in terms of CPU load increase and memory used and possibly build a PoC.
Explore the use of AI/ML tools to support software validation activities	Generation of tests, validation of technical specification, generation of test specification, analysis of results, etc.
Specification of an On-board Logging Service	Objective: Define the services to be provided by the On-Board Logging Service as included in SAVOIR OSRA Execution Platform. - What parameters should be traced, what configurability is needed for the tracing - What to do with the generated logs (dump them on a serial link or store them for later retrieval) - Defines ways to process logs, mainly to produce statistics - What tra cing mechanisms are to be supported (software tracing, hardware tracing, hybrid tracing)

Update Edisoft's RTEMS Timeline Tool to support RTEMS SMP	The main objective is to update existing Edisoft's RTEMS Timeline Tool, currently supporting only RTEMS Improvement, to support RTEMS SMP Tracing Service and multi-core architectures. This would help RTEMS users to observe the execution flow of RTEMS applications.
Trade-off of existing Linux observability tools	Ftrace, eBPF, Syslog, Journald, Collectd, Strace & Ltrace, LTTng, Auditd
Trade-off of existing PikeOS observability tools	Comparison between hooks and Instrumentation and Monitoring service
Analyse and implementation of auto-tests in OBSW	Main objectives: - Analyse the use cases where auto-tests would be valuable (reducing test time, providing more insight information in case of anomaly, etc.) indicating the capacities needed by the auto-test (execute TCs, if-then-else logic, loop logic, etc.) - Specify a dedicated language (or reuse one if exists) for auto-tests providing the needed capabilities - Specify auto-test engine - Implementation of both language and engine (if no use of an existing solution is deemed possible)
Generation of a HW/SW engineering handbook	Main objectives: - Define HW/SW co-engineering process during the whole system lifecycle (from design to maintenance) - Trade-off of tools that can help during the HW/SW co-engineering process (MBSE, High-Level Synthesis, co-simulation, etc.)
Analysis of observability mechanisms on multi-core based systems	Main objectives: - Analyse what specific parameters need to be traced to help with multi-core development/validation - Analyse how to adapt the design of the On-Board Logging Service to a multi-core platform
Analysis of improve of test means	Main objectives: - Analyse ways of improve test means used nowadays on projects - Study proposed improvements between others to be identified by the activity (increase the use of hardware probes foresee a trace logic analyser, dedicate a serial communication link to dump real-time traces, enable big data in test benches)

Analysis of improved validation methods	Main objectives: - Analyse ways of improve current validation methods - Study proposed techniques between others to be identified by the activity (on-the-fly analysis of TMs, add checks on logged data (outside TMs), add stress/robustness/endurance/operational tests, fault injection, validation of third-party software components)