Improved functional validation of on-board software by operational simulators – lessons learned from 15 years in-flight experience on Mars Express

Martin Shaw(1), Michel Denis(2), Mike Irvine(3)

(1) Telespazio VEGA Deutschland GmbH

c/o ESA/ESOC, Robert Bosch Str 5,

64293 Darmstadt, Germany

Email: martin.shaw@telespazio-vega.de

(2) ESA(ESOC)

Robert Bosch Str 5,

64293 Darmstadt, Germany

Email: Michel.denis@esa.int

(3) Telespazio VEGA Deutschland GmbH

Europaplatz 5,

64293 Darmstadt, Germany

Email: michael.irvine@telespazio-vega.de

INTRODUCTION

Owing to their high level of fidelity, increasing level of complexity and general ease-of-use, operational simulators are increasingly being seen as valuable tools for functional validation of on-board software (OBSW), despite their prime objective being as a training and preparation tool for the mission operations team.

Several anomalies or potential failures – identified in the course of 15 years of in-flight operations on Mars Express (MEx) - have been directly or indirectly attributed to limitations in, or functionality of, the OBSW. These have often resulted in the need to apply in-flight software (s/w) modifications, or to define operational workarounds. This paper presents observations based on a number of such cases, principally focusing on the following domains:

a. Mitigation of hardware deficiencies or limitations, for example, field programmable gate arrays, or possible S-band receiver failure;
b. Correction of software bugs, for example in the domain of the AOCMS failure detection, isolation and recovery (FDIR), or in the solid state mass memory (SSMM). To date, nearly 60 patches have been applied to the flying avionics s/w on Mars Express, the majority of these being to correct bugs identified either shortly before launch, or in-flight;
c. Refinements of OBSW usage resulting from an evolution of the operations concept of the flying mission;
d. Extensions of the OBSW to incorporate the use of on-board command procedures (OBCP's).

This paper identifies, for each domain, how operational simulators have actively aided the initial identification, and subsequent correction or potential mitigation, of such anomalies. Lessons learned from these experiences have been

drawn, and can be used to identify improvements both in future simulator design/usage, and also in the methodologies and timeliness by which such simulators can be used in the future, to offer improved functional validation of the on-board software design prior to launch.

THE BENEFITS OF OPERATIONAL SIMULATORS

The value of operational simulators in offering a mechanism by which to identify possible deficiencies in the OBSW, and to validate means to correct or mitigate them, arise because they:

1. Offer high levels of fidelity with respect to the flying mission (specifically equivalence to the full avionics s/w operating in-flight);
2. Offer a high level of complexity, in supporting in full fidelity the FDIR, and the interactions between the prime s/w components of the mission;
3. Support realisation of operational situations equivalent to those experienced on the flying mission (of particular value when the OBSW usage has to evolve to take account of changes in the operational profile of the mission);
4. Support full integration with the corresponding ground segment elements (e.g. Ground station contact periods, full interaction with the operational mission control and possible automation systems), which is often of value in validating the operational implementation of the OBSW changes;
5. Allow simulations over extended elapsed times, thus mimicking soak tests which would otherwise be impractical in an environment which can only support execution in normal time;
6. Are generally easy-to-use.

[Note: This does not devalue their prime benefit as a training/preparation tool to support a mission operations team.]

In the next section we offer three case studies which illustrate primary domains in which an operational simulator has been of critical importance on MEx.

CASE STUDY 1: MITIGATION OF H/W DEFICIENCIES

Example: Mitigation of a possible S-band receiver failure

Context: A failure of the primary S-band receiver, coupled with safe mode entry (for any other reason), if occurring during an eclipse, could lead to any attempts to restore commanding being over-ridden by mutually exclusive actions to reduce the power load in the eclipse. Ultimately, this would result in an inability to command the spacecraft, and eventually in multiple safe modes and potential mission loss within 46 hours.

Mitigation: The OBSW timers were modified significantly to increase time before a failure in the downlink receiver chain is detected, the timings being defined to optimally match the communication profile of the mission.

How was the operational simulator critical to this functional validation? It…

1. Supported implementation/validation of the OBSW change, by injecting a receiver failure;

2. Allowed test execution in real-time, simulating anomalous behaviour over time periods covering 5-10 hours;

3. Supported s/w validation over the maximum possible (70 day) timescale applicable for this s/w processing FDIR scenario, by modifying the running simulator time and thus allowing evaluation of the proposed s/w change in soak-test conditions (mimicking all the corresponding time-outs periods utilised in the receiver FDIR);

4. Allowed validation in both RAM and EEPROM (the latter being validated by imposing a processor swap).

CASE STUDY 2: MITIGATION OF S/W BUGS

Example: SAM timeout error

Context: This error was detected in the Venus Express launch campaign (but was also identified as applicable to MEx). It resulted from a weakness in the AOCMS FDIR, which could result in non-detection of a Sun acquisition sensor (SAS) failure, the available FDIR surveillances being too short to trigger (as durations in the concerned AOCMS phases were too brief). This failure would result in a processor reboot, which would fail to correct the problem since this would reconfigure onto the prime AOCMS units (including the failed SAS). The resulting failure to converge to Earth pointing mode would result in a spacecraft stuck in Sun acquisition mode, with multiple (>15) safe modes leading to **potential mission loss**.

Mitigation: The potential error situation was mitigated in three ways:

1. Introducing a new counter of the time spent in all AOCMS sub-modes;
2. Introducing a check if this time exceeds a fixed limit, and if so trigger an AOCMS hardware reboot onto redundant units;
3. Ensure consistency of all FDIR timers/triggers , with values appropriate for the maximum anticipated eclipse durations which will be encountered for the remainder of the mission (when no SAS output would be expected) – 100 mins was selected.

How was the operational simulator critical to this functional validation? It…

1. Supported implementation of the OBSW change by applying the appropriate patches (3 being needed to both the data handling and AOCMS processors) to the running s/w, and ensuring no unexpected and unforeseen conflict with the other running s/w tasks;

2. Supported validation by injecting a prime SAS failure and observing the expected subsequent behaviour of the OBSW after modification;

3. Allowed test validation of all FDIR triggers/timers modified in the OBSW change, by allowing simulation of conditions in which all timeout conditions could be encountered;

4. Allowed validation in both RAM and EEPROM (the latter being validated by imposing a processor swap).

CASE STUDY 3: PERFORMANCE IMPROVEMENTS ARISING FROM REFINEMENTS OF THE OPERATIONS CONCEPT

Example: The migration to file-based operations

Context: Weaknesses to data corruption in the solid-state mass memory (SSMM) resulted, in 2011, in a succession of safe modes, the ultimate mitigation of which was to isolate the normal mission time line (MTL – as stored on the SSMM), and migrate to the use of a short MTL (stored in the central data handling processor). Unfortunately, this migration resulted in the reduction of capacity in the available timeline from 3000 (Nominal) to 117 (short) commands [1]. This therefore required:

1. A significant reduction in nominal mission commanding volume;
2. A refined operations concept to store commands in dedicated command files, allowing all the commands which require to be executed in routine operations to be encapsulated within such command files, these being loaded/executed when needed, and
3. The development of a scheduling program to manage execution of these files, populating the short MTL with the required commands (or OBCP's) shortly before they were required to execute.

Mitigation: Realising this performance enhancement was achieved in three ways:

1. By the development of OBCP's to support all typical operational commanding activities (this realising a reduction of a factor ~4-6 in most payload instrument operations, for example);
2. By modifying the mission planning system to support the use of command files for command storage;
3. By the development of mechanisms (via OBCP) to manage the unpacking and execution of the command files at the appropriate times.

How was the operational simulator critical to this functional validation? It…

1. Supported implementation of all the OBSW changes required (for example, re-activation of OBSW functionality to support the use of OBCP's, which was not originally the mission baseline for MEx);

2. Supported validation by simulating creation, unpacking and subsequent execution of the dedicated command files, scheduling them for insertion into the MTL at the appropriate time, and confirming subsequent execution of these commands from the MTL when expected;

3. Allowed validation of all the corresponding modifications in the mission control system (several of which were necessary to support the verification of command insertion into the MTL, and their subsequent execution);

4. Supported all validation steps in real-time in a fully realistic (on-board and on-ground) operations environment.

OTHER DOMAINS IN WHICH AN OPERATIONAL SIMULATOR HAS HELPED THE FUNCTIONAL VALIDATION PROCESS ON MARS EXPRESS

Throughout the lifetime of the MEx mission, the operational simulator has helped in a number of other ways to provide functional validation of the OBSW, ultimately resulting in the development of a number of additional dedicated s/w patches or operational workarounds. Further examples of these are provided below:

Examples of s/w bugs in the SSMM corrected by dedicated patch:

1. Low-level pointer handling during read/writes (especially after pointer wrap-around);
2. High speed link interface errors between payload instruments and the SSMM;
3. Non-clearance of an error status after file transfer failure, blocking subsequent file transfers (which utilises intermediate files stored on the SSMM);
4. Read of TM packets of a specific length (i.e. multiple of packet size);
5. Failure to terminate data reads (leading to subsequent read de-synchronisation).

Examples of SSMM performance issues requiring operational workarounds:

1. Avoidance of disc fragmentation (to improve unit performance) which – though originally unforeseen - adversely impacted the refined operations concept, which required the need for regular creation and deletion of command files;
2. Termination of reads from the solid state mass memory, which were necessary to free resources and optimise data retrieval from the SSMM;
3. Perform cyclical clearance of the MTL intermediate files using a dedicated application programme, which was necessary to avoid unit non-performance and subsequent mission interruption, resulting from the blockage of tasks with a similar priority to those managing the MTL population and re-filling.

LESSONS LEARNED - WHAT DO OPERATIONAL SIMULATORS OFFER IN THE OBSW FUNCTIONAL VALIDATION PROCESS?

Operational Simulators are uniquely able to support full (end-to-end) functional validation of the OBSW, because they offer:

- The ability to execute long-duration soak tests with the running OBSW under realistic operational scenarii, for example supporting:
    - Continued creation/deletion of files under the most extreme operational conditions in the routine operations phase;
    - Continued filling of on-board storage files (i.e. packet stores) at variable rates, taking account of all possible instrument operational conditions;
    - Depletion and re-filling of the MTL with a realistic commanding profile.
- Validation of all branches of critical OBSW elements by supporting the execution of test scripts which mimic certain (pre-defined) critical contingency situations;
- The ability to inject a diverse range of contingencies, and in a controlled manner (reflecting how such errors may manifest themselves during real operations);
- Modelling of payload instruments to simulate the data flow (and data characteristics, such as packet sizes) anticipated in all routine operational mission phases;
- Functional validation of the running OBSW in an environment fully representative of that used in-flight.

POSSIBLE FUTURE DEVELOPMENTS IN THE APPLICATION OF OPERATIONAL SIMULATORS IN THE OBSW FUNCTIONAL VALIDATION PROCESS

Looking to the future, there is a growing trend for increased (modelling) fidelity within operational simulators, further strengthening their ability to support functional validation of the OBSW. This includes more detailed functional simulation of spacecraft units in general, offering higher fidelity environmental modelling where required (especially in the areas of spacecraft dynamics or thermal sub-systems), and in particular the wider use of processor emulators executing the real flight software beyond their traditional use in the avionics domain.

We cite two examples illustrating this trend:

- The BepiColombo operational simulator executes the SIMULUS (ERC32 processor) emulator at the core of the on-board computer, the failure control electronics, and the SSMM sub-systems, and includes hot redundancy of the processor modules within each sub-system.

- The ExoMars TGO simulator can run both hot redundant (ERC32 processor) emulators within the concerned platform sub-system, together with a LEON2 processor within the data handling sub-system and a LEON2 processor within the entry and descent module sub-system.

The wider use of processor emulators not only increases the realism and fidelity of the individual simulator sub-systems, but also the interactions between them via accurate models of their respective interfaces (be they data buses, SpaceWire routers etc.).

ACKNOWLEDGMENTS

The authors acknowledge both present and past members of the MEx flight control team for their creativity in defining (or identifying) deficiencies in the OBSW which required development (and testing) of software change(s), or which required mitigation by an operational workaround.

The operational simulation development and support teams in Telespazio VEGA are thanked for their respective roles in the provision of an excellent operational simulator for the Mars Express mission, of sufficient fidelity and complexity to realise the benefits identified in the present paper.

REFERENCES

[1] Lakey, D. et al, “From drawing-board to on-board: A new mission timeline on Mars Express via OBCP,” AIAA SpaceOps. Conference Proceedings 2014, Pasadena, USA.