Testing and Validating Operational Spacecraft Simulators

Marta Pantoquilho[(1)]
[(1)]*European Space Agency, European Space Operations Centre, Mission Data Systems Division*
Robert Bosch Strasse 5, Darmstadt, Germany
Email: Marta.Pantoquilho@esa.int

INTRODUCTION

The Mission Data Systems Division (OPS-GD) at the European Space Operations Centre (ESOC) is responsible for the specification of the requirements, the management of the development and the validation of all Mission Data Systems composed mainly by the Main Control System, the Spacecraft Operational Simulator and the Mission Planning System for the European Space Agency's missions. A new set of Mission Data Systems is developed for each mission based on common reusable infrastructure software.

The main user of the Mission Data Systems is the Flight Control Team of each mission. The Flight Control Teams at ESOC start flight operations training and preparation on a specific Mission through, mainly, the preparation of flight operations procedures while the spacecraft itself is being built and is therefore unavailable for testing. The Flight Control Teams training ends just before the mission launch and has its peak during the Simulations Campaign, a specific training that starts ca. 6 months before the launch. The Operational Spacecraft Simulator, one of the Mission Data Systems, is used by the Flight Control Team, mostly together with the Mission Control System, in order to surpass the lack of a real spacecraft with which to test procedures and train. Each Operational Spacecraft Simulator closely models one or more (in the case of constellations of satellites) Spacecraft with its systems and subsystems mimicked to the detail, including an emulator that runs the real binary of the Onboard Software.
The development, testing and validation of the complex Operational Spacecraft Simulator poses several challenges. The first and major challenge resides in the fact that the majority of the requirements on which the development is based must be generic enough to cope with reduced available Spacecraft specifications and knowledge at the time the requirements are specified. This derives from the fact that the Spacecraft itself, its Onboard Software and Spacecraft database, containing all the Spacecraft Telecommands and Telemetry, are also under development at the same time as the Operational Spacecraft Simulator. For this same reason, the likely changes that might occur in either of them (the Spacecraft systems and subsystems specification, Onboard Software or Spacecraft database) will result in a respective change in the Simulator models or in a new integration of the most recent Onboard Software and Spacecraft database. As a consequence, this requires a respective adaptation in the test and verification plan.
The testing and validation process of Operational Spacecraft Simulators must take into account all these dynamic aspects of modelling innovative, complex and under development Spacecraft.

This paper intends to illustrate exactly this complex process and lessons learned in testing and verifying Operational Spacecraft Simulators at ESOC, coping with the parallel development of the spacecraft being modelled and the consequences that it brings.

OPERATIONAL SPACECRAFT SIMULATORS

A Spacecraft Operational Simulator is a complex piece of software composed of a high fidelity model of the real (and extremely complex) Spacecraft (S/C) and its ground segment interfaces. Its main objective is to provide a representative simulation of the spacecraft platform and payload control, by inclusively running (using an emulator) the real software (without any modifications) that will run on board the S/C. This is called Onboard Software.

The behaviour of the spacecraft (platform and payload) must be modelled to the extent that to the Flight Control Team (FCT) the Telemetry production is indistinguishable (as far as possible) from the real Spacecraft [3]. The ground stations and network interfaces that allow a direct connection to the Mission Control System (MCS) must also be modelled in the Operational S/C Simulator.

In addition, the simulator must support the injection of failures (predefined in the requirements) by the user in both the S/C model and the ground segment model [3].

The lifetime of an Operational Spacecraft Simulator is the same as its mission's planned lifetime.

Involved Teams

The development of any Operational S/C Simulator (throughout the rest of the text "Simulator" will be used indistinctly of Operational S/C Simulator holding same meaning) requires the involvement of several teams: The supplier team, the customer, the information providers and the operators and users.
The Simulator supplier team is composed of the infrastructure supplier, and the models' developer team (industry) which develops the S/C models and integrates them with the used infrastructure.
The customer of the Simulator is the Mission Data Systems Division, OPS-GD, at ESOC.
The information providers team includes at least the System Engineering Team, located at ESTEC, and the equipment supplier teams composed of the industry partners responsible for the construction of the S/C itself.
Finally, the Simulator operators and users team is composed by the S/C Flight Control Team members and the Simulation Officers assigned to that particular mission.

Development Challenges

The development of a Simulator is a challenging task due to the combination of a set of factors:

a. The S/C to be modelled in the Simulator is very complex.
b. The knowledge ESOC has about the S/C is distributed and divided through a reduced set of people (the FCT), each containing deep knowledge about only a subset of systems of the S/C.
c. The S/C is often being developed in parallel to the Simulator and thus its specifications are evolving with the development and the Simulator has to reflect these changes (see Fig.1).
d. The specifications of the S/C, required to develop the Simulator, are provided by external entities, the S/C developers, which are not immediately available to provide them, or to clarify doubts.
e. The S/C onboard software (OBSW) and SDB (Spacecraft Database containing all the Telecommands and Telemetry) are also being developed in parallel to the Simulator by external entities. New versions of this software must be integrated into the Simulator as soon as they become available. The evolution of the onboard software allows testing additional functionalities of the Simulator models that require an interface with the onboard software (see Fig.1).
f. Long before it reaches the end of its development phase, the Simulator must fulfil the needs of Mission Control System (MCS) testing, flight control procedures validation, flight control team training and simulations campaign preparation (see Fig.2).
g. The latest version of the infrastructure software used at ESOC (SIMULUS) that offers the common functionalities to most Simulators, and is integrated in the Simulator, is sometimes a new product that didn't have the chance to be used operationally. Since this software has not reached the end of its development, it can be an additional source of uncertainty and potential problems in the development of the Simulator.
h. It is software that is used operationally.

These items (S/C specification and design documentation, S/C Database, S/C onboard software) are provided to the industry (that will develop the Simulator) by ESA. Due to its strong dependency from these inputs, the Simulator development and testing is strongly influenced by the S/C development stage.

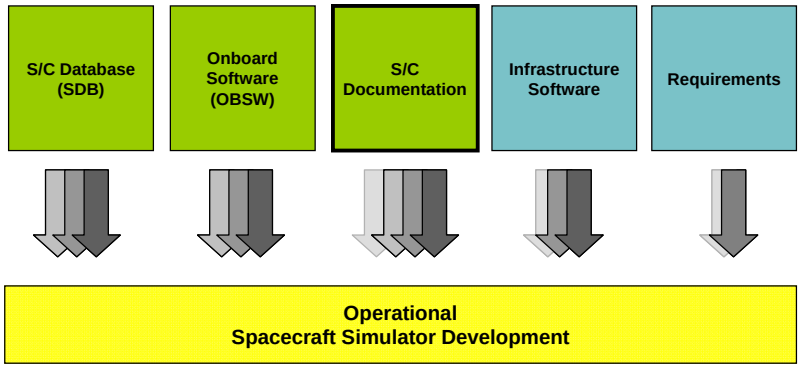

Fig.1. The main inputs to the development of Operational S/C Simulators. Key: Green: inputs provided by external entities. Blue: inputs provided by internal entities. Arrows: illustrates the amount of versions of each input.

Although the relevance of the Operational S/C Simulator is significantly higher before launch, after launch, and after the development of the Operational S/C Simulator is finalized, the Operational S/C Simulator can be used to re-train the FCT, to validate procedures, to simulate and verify future manoeuvres (e.g. Comets or Planets flybys), to validate OBSW patches and to investigate onboard anomalies (see Fig.2) [3].

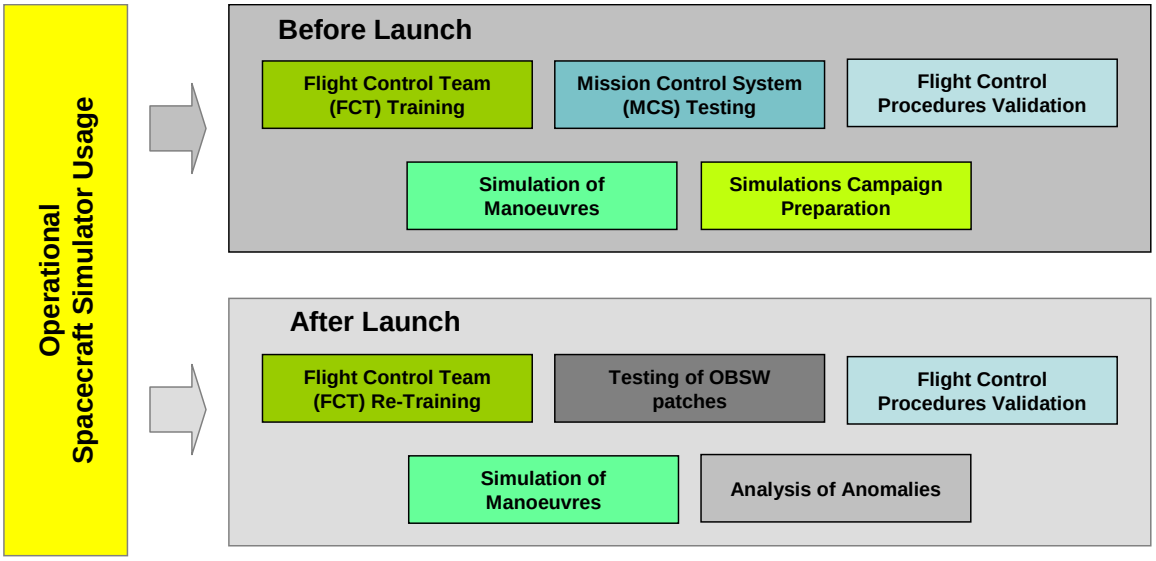

Fig.2. Multiple usage of each Operational S/C Simulator before and after it reaches the end of its development (which usually coincides with the mission launch).

DEVELOPMENT APPROACH

In order to cope with most of the above mentioned challenges, ESOC has adopted a flexible development approach [1,2] based on frequent incremental deliveries, the reuse of the common infrastructure software and generic requirements ready to cope with S/C specification changes, uncertainties in information, increased complexity of the S/C, the criticality of the software and the demand associated to a short time for implementation (due to the FCT needing the Simulator to always reflect the evolving design of the real S/C).

Generic Requirements

The development of an Operational S/C Simulator starts typically ca. 3 years prior to the expected S/C launch and finishes a few months after launch. The requirements of the Simulator are written early after the spacecraft starts being developed. This results in requirements generic enough to cope with the limited available information on the S/C and any potential changes in the spacecraft specifications. Usually the requirements specifications for Simulators act as guidelines on how to model the behaviour of the spacecraft. They usually state expected accuracy, deviation and simplifications that can be made between the Simulator and the real S/C, expected performance and robustness of the software, as well as generic requirements with a generic implementation (e.g. any parameter should have functions to force its value, to check if it's out of limits, and to add noise)[1,2].

Reuse of Infrastructure Software

S/C usually have little in common among each other, besides the general ground communications. The exception to these are the S/C that belong to the same Mission (where all S/C are identical and meant to fly e.g. in formation), or reruns of a previous mission (e.g. Cryosat 2) [2]. Nonetheless, the structural bone of any Simulator is common to all S/C Simulators. These common components are called the Infrastructure, and are reusable across S/C Simulators. The Simulator infrastructure in ESOC is composed of the following components:

- SIMSAT (Software Infrastructure for Modelling Satellites) provides the simulation kernel and generic MMI.
- Generic Models, including a spacecraft Positioning and Environment Model, a spacecraft Dynamics Model, a Packet Telecommand (TC) and Telemetry (TM) Toolkit (SIMPACK) implementing the ESA Packet TM and TC Standard, and software emulators of onboard processors.

- A Ground Station Simulation System (or Ground Models in Fig. 3), including models of the TC encoder and TM decoder ground-station equipment, implementing the space link and network link for both the ESA Stations Protocol and the CCSDS Space Link Extension (SLE).

At ESOC, there is a dedicated section responsible for the development and maintenance of infrastructure software (OPS-GI). The infrastructure software must accompany the increasing needs of Simulator requirements, the approved standards as well as the hardware's evolution. Therefore, the infrastructure software is also evolving software that is being, most of the times, improved and extended at the same time as the Simulator.

To minimize the risks every development of an operational Simulator starts by integrating the latest available (and completely) developed infrastructure software. However, sometimes the S/C requirements demand that the Simulator development takes into account the integration of new, not yet finalized, versions of the infrastructure software (e.g. due to envisaged hardware and Operating System upgrades).

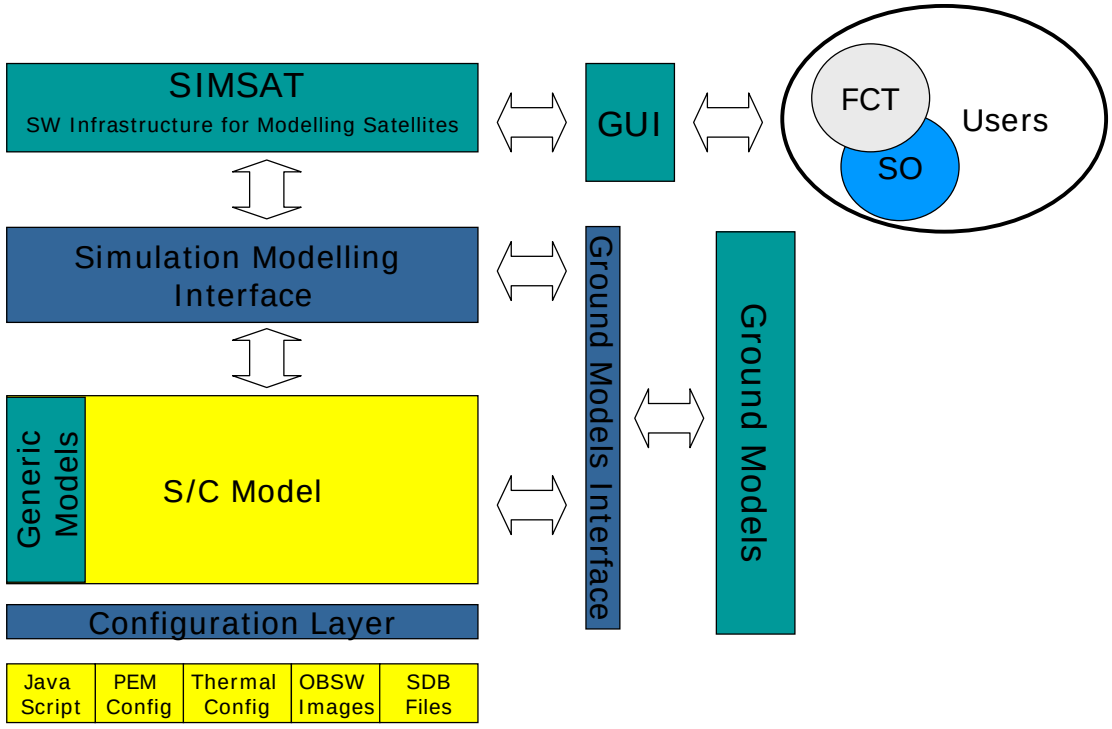

Fig. 3. The simplified architecture of an operational spacecraft simulator based on ESOC's infrastructure.

Incremental Deliveries

It is also part of the flexible approach followed at ESOC to schedule several incremental deliveries. Each delivery contains more functionalities implemented, leaving for last the systems and subsystems that are prone to suffer the most changes or lack the majority of information, due to their novelty or complexity (e.g. the payloads). Typically the first delivery of an operational Simulator serves to prove that the underlying chosen infrastructure software integrates seamless together, and that the common functionalities provided by that software operate correctly. Posterior deliveries will progressively integrate the S/C systems models and their functionalities and also the newest versions of the onboard software.

Each delivery is followed by a period of typically 4 months dedicated to the complete testing, verification and validation of the delivered software. During this time, the parties responsible for testing prepare or update the previous test plan, execute the tests, and raise the defects when necessary. In case the defects are considered critical and urgent (meaning they are blocking further use of the Simulator), new patch deliveries to fix those defects are usually agreed between ESA and the developers.

Impact on Testing

The flexible approach used at ESOC for the development of S/C Operational Simulators has proved to be efficient and effective, reducing the cost of development, reducing the overall time for development and, more importantly, reducing the time for the first delivery usable by the FCT.

This strategy increases the flexibility and provides for better adaptation to changes. However, it has an impact on the testing, verification and validation process of Simulators.

Generic requirements increase the difficulty in specifying test cases and add complexity to their own verification. In addition, when the spacecraft specifications are made available, become more precise or even change, adaptations in the Simulator models are required. Thus, the complexity of the testing and validation process that must accompany all the updates increases.

The flexibility of being able to integrate new, not finalized, infrastructure software, has also an impact on testing the Simulator. The Simulator will be affected by the undetected defects of infrastructure software that has been tested but not integrated in a real operational Simulator (which has more demanding performance and usability requirements than what is validated in the infrastructure test environment). In these cases, the testing, verification and validation of the Simulator will also contribute to the testing and signalling of defects of infrastructure software. The major concern comes when the infrastructure software defects are blocking the testing of the Simulator. Since, at ESOC, the infrastructure software is developed by an independent section, its lifecycle is not (and cannot be) aligned with the development and schedule of the several Simulators that are being developed at the same time at ESOC.
Any defects in the infrastructure software must go through their own independent lifecycle until they are corrected and provided to the users (in this case the developers of the Simulators). Defects that are blocking the testing of the S/C Simulator, however, might contribute to delays of the S/C Simulator testing, and consequently to delays in its development.

TESTING, VERIFICATION AND VALIDATION

Testing and validation of software products are time and resource consuming tasks. In the case of Simulators the cost and complexity of testing is greatly inflated by the extreme complexity of the system, the reduced set of resources with knowledge about the S/C required to write meaningful test cases, the short development time, the unavailability of detailed S/C documentation and the potential changes to the S/C specifications and documentation. However, because testing is a critical and essential task to ensure that S/C Operations can be carried out, it must be done systematically, effectively and efficiently throughout the whole development cycle, by representative members involved in its development, specification, management, and usage.

Testing Lifecycle

At ESOC, OPS-GD manages the Simulator development contract and is also responsible for the first level testing and validation from a software point of view. This team has an appointed Technical Officer (TO) that manages the contract and helps the FCT to make sure they will get the system they need. The task of testing the system is shared between the TO, the FCT and to a lesser extent the Simulations Officer (SO). The TO is responsible for the first validation, before handing the system over to the FCT and should detect any major software flaws, and not covered requirements. The FCT's task is to perform the nominal operational system validation and testing, at the level of detail that only the S/C experts are able to. The tests performed by the FCT include separation sequence, manoeuvres, asteroid and planet flybys, etc.
The SO task is to test the specific functionalities required during the Simulations Campaign (e.g. failures, forcing of values) and test the non-nominal behaviour of a S/C (when subject to failures and anomalies).

The FCT is obviously the major player in the validation of the system, since the TO does not possess the same knowledge of the S/C as the FCT members. The software life cycle is organized such that the validation and testing can be phased and shared between FCT, TO and SO. After each incremental delivery of the S/C Simulator, during one month, the TO installs the software and performs high level testing and validation of the software. If the software is

considered acceptable by the TO, it is considered Provisionally Accepted and is then passed on to the FCT during three months for more specific tests. The validation of the infrastructure software is also part of a TO tasks.
During these three months the FCT (and the TO to a much lesser extent) perform operational validation tests and document all identified defects. At the end of the 4 months, both the TO and the FCT produce independent Test Reports and if it is considered that the software implements the agreed requirements for that delivery and follows the S/C specifications for the systems modelled and implemented in that delivery, then the delivery is considered accepted. The SO usually becomes involved in the testing later on the stage of the development, since the first deliveries usually do not contain the functionalities that he/she is responsible for testing. Fig.4 below depicts the testing cycle.

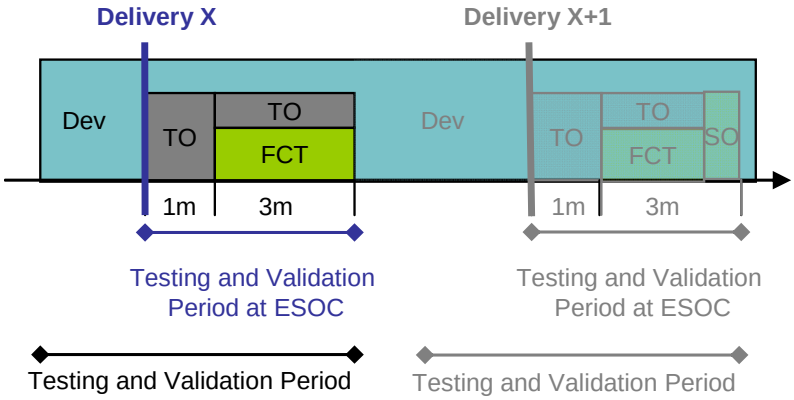

Fig.4. The sharing of the testing and validation period of each Operational S/C Simulator incremental delivery. Key: TO: Technical Officer, Dev: developing team, SO: Simulations Officer, 1m = 1 month, 3m = 3 months.

Starting approximately six months before launch, lasting until a few days prior to launch, the Simulation's Campaign, lead by one or more Simulation Officers takes place. In fact, the Simulation's Campaign is the period in which the Simulator is strongly stress tested and in which extreme conditions of the models are tested. By the time the Simulations Campaign starts the Simulator has already been heavily tested and used. Therefore, this is the last period of intensive Simulator testing and fine tuning before the warranty period ends, and the Simulator enters a maintenance phase. As such the Simulations Officer has also a big role in the Simulator's testing.

The persons involved in testing the Simulator (and their tasks) are the following:

- The Technical Officer (TO) is responsible for managing the software development contract, to accept the deliveries, install them, perform software high level tests an verify the coverage of requirements.
- The Flight Control Team (FCT) members possess the most detailed information about the S/C and are best suited to perform operational system tests and verify if the implementation in the Simulator corresponds to the real S/C. Each FCT member is specialized in one or more specific S/C systems, and is therefore responsible to test the systems that he/she knows best.
- The development engineers and testing engineers that compose the testing team are the first testers of the system, being able to test the actual software implementation. However, due to their limited knowledge in the S/C itself, their tests verify usually mostly the software robustness and performance.
- The Simulators Officer (SO) is responsible for the preparation of the simulations campaign and the training of the FCT. His usage of the simulator allows for stress testing, testing of failures and of boundary conditions.

The Developers, the TO and the FCT all have their own independent test cases and scripts. All of them file in an independent report with the results of their testing and verification process.

Testing Strategy

A test strategy provides the overall direction for the software testing needs of a project. Developing a test strategy is about setting direction and resolving high-level testing questions. As the system progresses and more S/C information becomes available, more detailed test plans, test cases, and test scripts can be prepared.

The S/C specifications documentation is an essential key element in all phases of the Operational S/C Simulator development, including testing and validation. The development is done based on the S/C specifications available at that time, and so it the testing. Generation of test cases and scripts must be done accordingly. It is not rare to have updates in documentation during the same development phase, e.g. while a delivery with certain functionalities is being prepared. It can also occur that the testing and validation is done using a different (more updated) documentation version than the one used for the development.

The testing strategy must take into account the percentage of requirements coverage that the applicable standard demands (e.g. 80%, 90%) and assure that representative requirements from all types and from all the Simulator components are covered by the tests.

Testing Techniques

Testing techniques and tactics in the software engineering field are targeted at software developers or software testers. They do not target the end user and the role of the TO in system testing is not clearly contemplated in standard software testing techniques.

There are several testing techniques available for software testing. At the developer level there are the unit tests, used to test a single unit of code, the integration tests, that combine software modules or units into an effective test group and system tests (also called functional or application testing) that combine components into a complete system to verify a specified requirement.

The TO uses black box testing (or behavioural testing, that focuses on the functional requirements of the software) to perform the testing, including the system integration testing (between the Simulator and the infrastructure software).

The developers use automated regression and system tests based in scripts (usually written in javascript). These tests are usually passed on to the TO who re-uses them also to perform his own tests, in the deployment environment.
However the usage of automated tests does not fit all purposes of a Simulator testing:

- Testing of Man Machine Interface (MMI) functionalities is not done using automatic tests (it is a manual interaction process).
- Testing that requires many Telecommands which will depend on the S/C's state is usually also not done automatically, since it requires the interaction with the Mission Control System.

The tests performed by the FCT are mostly manual tests. Some scripts can be used to put the Simulator in the desired state (e.g. to simulate that the S/C is at a certain stage from its Launch and Early Orbit Phase), by saving a breakpoint in the Simulator and restoring it later. A breakpoint is a snapshot of the current state of all models in the Simulator. The creation of the breakpoint, at an early stage of the Simulator development, is done manually, but can be done automatically through a script in later stages (also to be noted is that the breakpoint restoration might not work between different deliveries versions). Most of the system verification tests require the usage of a real Mission Control System (MCS): in reality the S/C is commanded from the MCS, and this is how the FCT is trained to interact with it. Therefore in the Simulator tests, they use the Simulator as they would use the real S/C, i.e. by commanding it from the MCS, and checking the effects of their commands in the Telemetry produced, in order to assess the correct functioning of the Simulated S/C.
This process is, clearly, difficult to automate. It also has the disadvantage that it requires good knowledge of the MCS in order to perform the tests.

Future Improvements

At the beginning of a Simulator development the biggest challenge is the fact that high level tests require a long preparation, a deep knowledge of the S/C, and a very advanced status of Flight Operation Procedures (produced by the FCT on how to operate the S/C) and SDB. However stable versions of the above are only available very late in the testing process. This leads to the fact that only at this late stage (occasionally during the Simulation's Campaign) it is possible to properly test the Simulator in an operational scenario and this is where the major problems are detected. Nonetheless, in order to mitigate the lack of the Onboard Software in the initial deliveries of the Simulator, OPS-GD

introduced a Generic Emulation Test System (GETS) component as part of the first delivery of the Simulator. GETS has also been used in the System Testing of the Reference Architecture [4] in order to provide a generic, but still realistic test set-up. The Reference Architecture is used in the development of SMP-2 based (Simulation Model Portability 2 Standard) Operational Spacecraft Simulators [5]. GETS provides the means to produce a simple but generic and highly configurable version of the OBSW, which can be used in the early phases of the simulator development to reduce the dependencies on the delivery schedule and continuous changes of the spacecraft OBSW.

It is acknowledged that in order to develop the Simulator and produce proper test plans it is necessary to get the relevant S/C documentation on time for both the development phases and the validation phases. In addition, afterwards it is also necessary to be able to understand the S/C documentation properly, which can per se also be a challenge. A possible solution for these issues is a closer, with fast interactions, contact with the S/C prime and sub-developers, not only for the FCT but also for the Simulator developers and TO.

Currently FCT and TO perform the Simulator's validation and testing separately, through different test cases, using a distinct set of tools and producing independent assessment reports. Although this approach increases the diversity of the tests and allows for very different testing approaches to the same functionality, better coverage could be achieved through closer cooperation. In addition, the involvement of the SO at an earlier stage of the development could be very useful in testing the system in realistic and S/C contingency scenarios.

Automation in testing should also be increased in order to speed up the regression tests and their requirements coverage. ESOC is currently developing a tool to transform the Flight Operation Procedures in tests in order to easily perform S/C system tests from FCT specifications at a later stage of the Simulator's development. The same tool could also be used to automatically generate test scripts from the specified steps in test cases.

CONCLUSIONS

An Operational Spacecraft Simulator is a complex piece of software that must provide a high fidelity model of the real spacecraft, and is built on top of existing complex and sometimes new infrastructure software that has not yet been validated in a real Operational S/C Simulator.

At ESOC there is, formally, a dedicated section to procure and manage the development of these complex Simulators, test and validate them, OPS-GD. However, in practice, a more flexible approach has been put in place, where the testing and validation tasks are shared among Developers, Technical Officer (from OPS-GD), end users (the Flight Control Team) and finally the Simulations Officer. This decentralized testing and validation approach is not the silver bullet in what testing and validation are concerned, but is a practical and effective approach to cope with a very complex set of problems mainly resulting from the increasing complexity of the S/C to be modelled.

Nevertheless it is clear that the testing and validation process of Operational Spacecraft Simulators can be improved with the introduction of new testing techniques and procedures, such as increased usage of automated tests.

OPS-GD is always, actively, in the outlook for improvements to their approaches, and thus, it is foreseen that new approaches to testing will be tried out to cope with these extremely complex systems, and since experience is the best teacher in what concerns testing and validation, the lessons learnt will be fed back into the followed testing process in order to further improve it. The major requirement in any new approach is that it must be flexible enough to cope with the intrinsic characteristics and challenges of testing and validating Operational Spacecraft Simulators.

REFERENCES

[1] V. Reggestad, D. Guerrucci, P.P. Emanuelli, D. Verrier, "Simulator Development: the flexible approach applied to Operational Spacecraft Simulators", Proceedings of SpaceOps 2004 Conference, Montreal, Canada, 2004.
[2] Vemund Reggestad, "Spacecraft Simulator Development: Lessons Learned from the Cryosat Earth Explorer Mission", Proceedings of SpaceOps 2006 Conference, Rome, Italy, 2006.
[3] ESA-ESTEC Requirements & Standards Division, ECSS Space Engineering, "System Modelling and Simulation", Noordwijk, The Netherlands, 2007.
[4] European Space Agency, "SMP 2.0 handbook", ref. EGOS-SIM-GEN-TN-0099, October 2005.
[5] European Space Agency, REFA: Spacecraft Simulator Reference Architecture Specification, Vol. 1, ref. EGOS-SIM-REFA-SDD-1001, October 2010.