A cloud-based tool for automating SW testing in space projects

Workshop on Simulation and EGSE for Space Programmes (SESP)
26 - 28 March 2019

ESA-ESTEC, Noordwijk, The Netherlands

Massimo Tipaldi [(1)], Davide De Pasquale [(1)], Luca Germano [(1)], Massimiliano Di Penta [(2)], Fiorella Zampetti [(2)], Marek Prochazka [(3)]

[(1)] Intelligentia srl
Via Del Pomerio 7, 82100 Benevento, Italy
Email: {massimo.tipaldi, davide.depasquale, luca.germano}@intelligentia.it

[(2)] University of Sannio
Piazza Roma, 82100 Benevento, Italy
Email: {dipenta, fzampetti}@intelligentia.it

[(3)] European Space Research and Technology Centre
Keplerlaan 1, PO Box 299
2200 AG Noordwijk ZH, The Netherlands
Email: marek.prochazka@esa.int

ABSTRACT

SW Verification & Validation has a paramount importance for embedded, mission-critical systems. It is crucial to focus on test process definition and testing automation approaches from the early phase of any SW project. However, automating SW V&V activities can be difficult when the development process is geographically distributed and when simulators or physical environments are required. This paper introduces AIUTO (Aerospace Integration and Unit Test Organizer), an integrated multi-user cloud environment to automate the testing process for systems even in presence of geographically-distributed teams. Specifically, AIUTO supports the design, implementation, execution of tests, and facilitates test report analysis. AIUTO is currently being used by Intelligentia for the on-board software unit tests of Meteosat Third Generation satellites.

INTRODUCTION

Software Verification & Validation (SW V&V) assumes a paramount importance in the development process of any software system. This is particularly true for embedded, mission-critical systems [1, 2], where dependability is crucial to ensure the mission's success [3]. At the same time, studies have reported how SW V&V can account for up to 70% of the overall software development effort [4]. Such an effort can be reduced by introducing testing automation and, ultimately, setting up a Continuous Integration (CI) pipeline in which software is analysed and tested at every change [5]. However, SW V&V process may become particularly complex and error-prone when the development project is geographically-distributed and when, as it often happens for embedded systems, simulators (sometimes with limited licenses) and physical environments are required for testing purposes. Therefore, specific environments are required to support SW V&V in these specific contexts.

In this paper, we present AIUTO (Aerospace Integration and Unit Test Organizer), an integrated multi-user cloud-based application to streamline and automate SW test case design, implementation, and execution. AIUTO focuses on unit and integration tests, and has been conceived to support spacecraft On-Board SW (OBSW) verification activities. AIUTO can interact with different SW testing tools (e.g., IBM Rational Test RealTime (RTRT) or Cantata), and OBSW simulated target environments (e.g., mission-specific SW Validation Facilities), where the tests are actually executed.
More specifically, AIUTO provides the following features:

- Integration with SVN/Git software repositories;
- Web-based test design editor;
- Test script implementation, supporting third-party tools such as IBM RTRT or Cantata;
- Multi-node, parallel and sequential, deferred or online test execution (e.g. on distributed simulators) optimizing the usage of shared and expensive software licenses to multiple testers;
- Test results reporting and analysis in comprehensive web dashboards exportable in Excel sheets or via APIs to third party software.

AIUTO has been designed and developed as a hybrid cloud application with a particular focus on the security and confidentiality aspects of the source code to test. AIUTO can create a cloud-based testing environment in order to avoid unnecessary source code delivery to external suppliers and reduce security risks. The source code can remain at customer premises and can be exposed via a data-layer driver. The latter analyzes the structures of the SW modules under test and, depending on the assigned tasks, shares its parts such as packages, classes, functions and corresponding SW Detailed Design Document (DDD), diagrams, etc. The driver is then connected through secure and encrypted channels (e.g. SSL VPN) to the dedicated instance hosted within the Intelligentia's Cloud. Thanks to its architecture, external testers can also join the project for testing, even at single function level from any region of Europe or even in the World, by allowing virtually "infinite" scaling of the testing team in a purely technical fashion, while preserving security issues. AIUTO tool uses a gamification strategy in order to create competitions among testers for badge rewarding. AIUTO allows monitoring the Key Performance Indicators used to track and assess both the SW testing activity progress and the SW quality metrics.

The main goal for Intelligentia is to develop a sort of marketplace for qualified testers, where the external users can be rewarded according to the actual Source Lines of Code (SLOC) successfully tested and customers can allocate the budget for the overall SW testing phase based on such SLOC schema. AIUTO is being used by Intelligentia for the OBSW Unit tests in Meteosat Third Generation satellites, in particular for the SW application running on the platform on-board computer. Such SW application can also be referred to as Central SW (CSW).

The paper has been organized as follows. Sect. 2 provides an overview on spacecraft OBSW V&V activities and the reasons of using AIUTO for SW testing. Sect. 3 outlines the AIUTO architecture, while Sect. 4 focuses on the main AIUTO user-level capabilities. Sect. 5 concludes the paper and provides some future developments for AIUTO.

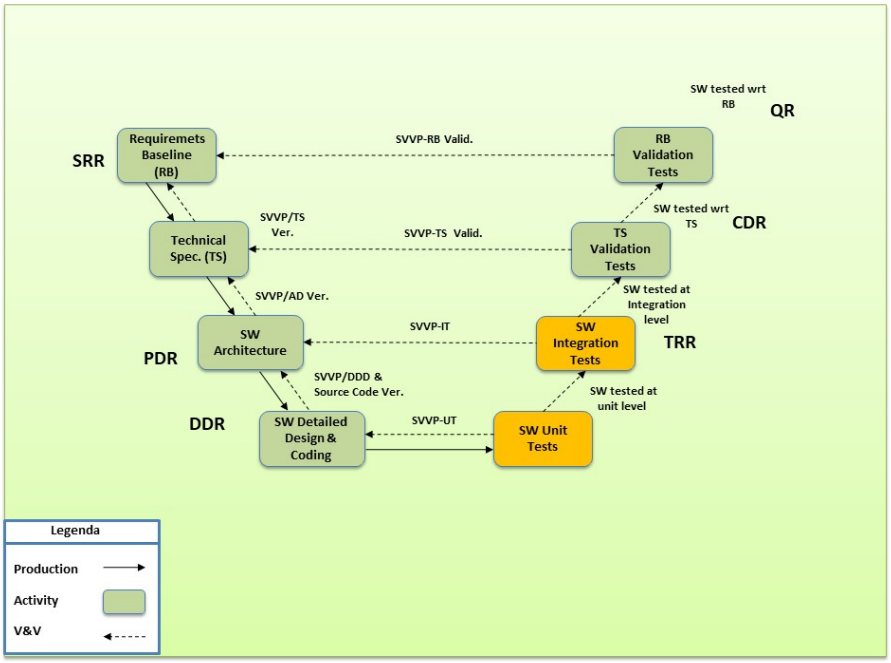

Figure 1 OBSW Development via a waterfall model

VERIFICATION AND VALIDATION PROCESS FOR CRITICAL SPACE SW

SW V&V activities are performed during the SW development. They aim at finding and removing errors that can have been introduced during SW development (and maintenance). As for spacecraft OBSW projects (and in line with the general trend in the development of SW critical applications), more and more importance is being attached to the SW V&V activities by all the project stakeholders. They are very expensive, and thus it is crucial to focus on test process definition, testing automation approaches, testing tool chain set-up from the early phase of any SW project (or even better at company level) [6]. Fig. 1 shows the typical spacecraft OBSW development process, which includes the SW V&V activities and is organised according to the waterfall model.

The ECSS standard ECSS-E-ST-40C [7] offers an exhaustive description of the SW V&V activities currently adopted in space projects. They are defined according to the associated SW criticality categories, ranging from level A up to level E, with A the highest level. Spacecraft OBSW is assigned to either criticality category B (i.e. “software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in critical consequences”) or criticality category C (i.e., “software that if not executed, or if not correctly executed, or whose anomalous behavior can cause or contribute to a system failure resulting in major consequences”) [8].

In this paper, we address SW Unit Tests (UT) and SW Integration Tests (IT). The former verifies that the functionality of each single method (i.e., a C function) works correctly as specified in the SW Detailed Design. Equivalence Class Partitioning (ECP) and Boundary Value Analysis (BVA) are used in order to conceive SW unit test cases for each C function. At least a representative set of nominal values, including the valid minimum and maximum, and the next value beyond each boundary (to cover invalid values) are addressed. During the SW UT, all the invocations of functions outside the source file under test are stubbed. Tests involving function calls that are located in a single source file are subject to SW unit test activities, see Fig. 2.

SW Integration Tests can be potentially much more expensive than SW UT. A pragmatic approach is usually considered for Spacecraft OBSW, which is in line with its typical criticality category B. SW integration tests are usually carried out in order to verify that the OBSW main component interfaces work correctly as specified in the SW Design Document, see Fig. 2. Such verification can be performed via testing or review of design. In this context, we can also consider numerical verification of the Attitude & Orbit Control System (AOCS) algorithm libraries both at component level and after their integration into the OBSW as well as the verification of the correct integration of third part developed SW components.

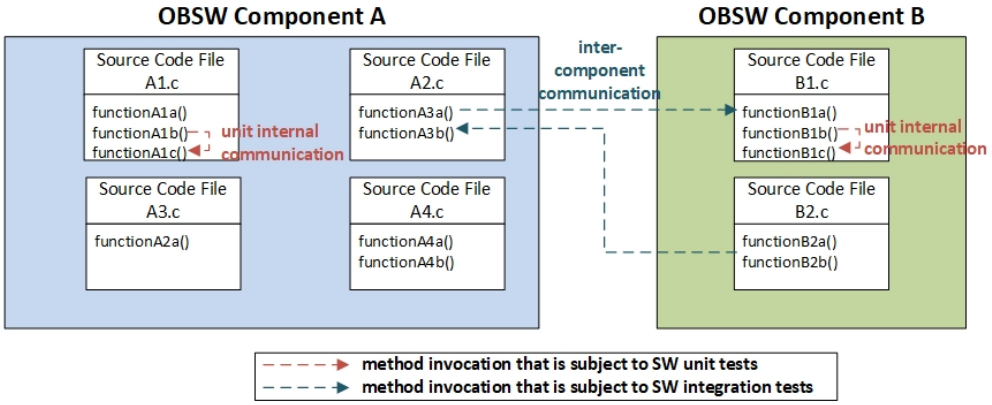

Figure 2 SW Unit & Integration Test approach

The following main activities are performed in the context of SW UT [7]:

- Set-up of the test environment and identification of the SW modules under test;
- Review of the corresponding SW Detailed Design, while documenting discrepancies by means of SW Problem Reports (SPRs);
- Specification of the UT cases (i.e., based on the SW Detailed Design and the testing approach, we have to specify the test cases for each C function under test. Each test case is mainly characterized by its input data, the expected results, and the verification criteria);

- Test procedure definition, including test script implementation for all the test cases;
- Versioning of the test case design/implementation over the different SW versions and baseline definition;
- Execution of the unit tests for the SW modules under test in the predefined test environment;
- Test result analysis, SPR production & handling (if necessary), and test report preparation.

Similar activities are performed for SW integration testing.

Justification of the proposed UT/IT approach via AIUTO

SW V&V activities are very expensive and are featured by many repetitive activities. For instance, after having specified the test cases, SW testing engineers have to spend a lot of time in coding lengthy and error-prone test scripts [6]. Existing tools (such as the IBM Rational Test Realtime) are able to process the OBSW source code files and generate automatically UT script skeletons, which are then used as basis for the test script implementation. However, such automated generated skeletons have to be heavily modified to incorporate the test case logic, and as for complex source code, this can result into coding script files with thousands of lines of code difficult to read and maintain. On the other hand, during the test case specification phase, it would be useful to have a tool supporting some steps e.g. the test input data definition and the pruning of redundant test scenarios. Furthermore, AIUTO can automatically generate test reports and it is able to support of test result analysis, such as the source code coverage figures and the traceability of work completeness per each function under test. In principle, AIUTO aims at automatizing the UT and IT activities by supporting the definition of the test case scenarios and reducing the gap between the test case specification and the automatically generated test scripts. Some features are currently available in the tool, others have to be consolidated.

AIUTO FUNCTIONAL DESCRIPTION & ARCHITECTURE

AIUTO is a web-application with no need for complex setup or configuration for each user. It is compatible with most of the used browsers thanks to the fact that it adopts the most advanced HTML5, CSS3 and JavaScript libraries. AIUTO has been designed for a hybrid cloud environment where e.g. the OBSW source files, the test artefacts, and the front-end application can be deployed in different Internet nodes, which can be linked through secure Virtual Private Networks, or through a dedicated network infrastructure. A test server can be also integrated to schedule automatic execution of a tagged release or nightly builds of software versions, therefore enabling a continuous SW integration process.

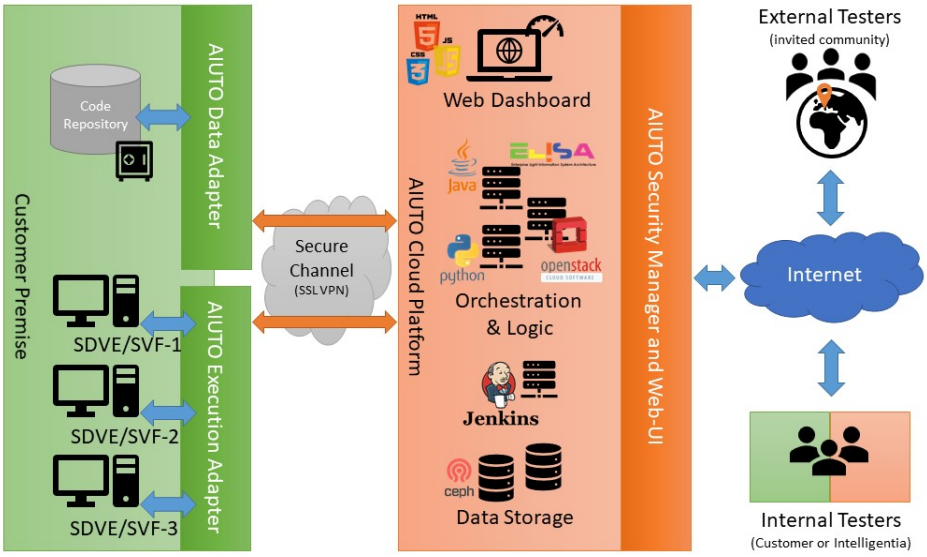

Figure 3 AIUTO architecture

Fig. 3 depicts the testing environment architecture. It is composed of the following distributed components:

- A cloud open-stack platform, on which AIUTO is installed within a Java Enterprise Edition distributed container, which provides the tools for the testers, the results dashboard, and the team control interface. This node can be replicated several times for scalability, business continuity, load-balancing. It can be common to several projects and it is the only entry point for external testers, unknown to the owner of the source code under test;
- The AIUTO Data Adapter (compatible with SVN and Git repositories) hosted at customer premise and that has to be allowed to "read" the source code to process it, and create the online data-model used by the AIUTO application. If in a black-box configuration, the source code is never transmitted to the cloud distributed database used as storage for AIUTO, but just the prototype of the function, the corresponding DDD and optional attached diagrams are shared with the external tool.
- Test cases are created in AIUTO by external testers via the web-interface, then the AIUTO Execution Adapter rebuilds the scripting file structure and store them locally for execution phase;
- One or more SW Development & Verification Environment (SDVE) servers, where the production code has been downloaded. The goal of the SDVE servers is to allow the compilation of the production code along with the tests. This node is hosted at customer premise (in order to use the customer licenses of qualified software test benches) and optionally this node can be installed at Intelligentia premise;
- A continuous integration server (Jenkins in our case), which polls the test cases from the storage system, deploys them on the distributed SDVE machines, where, as explained before, they are compiled with the source code;
- Once the source code and the tests have been compiled on the SDVE, they are deployed on the SW Validation Facility (SVF) and executed there. After that, results are collected and sent back on the tester's personal dashboards and visualized in the web reporting dashboards.

AIUTO IN ACTION

In this paragraph, we focus on the main AIUTO user-level capabilities. We present them by considering the usual UT/IT work-flow and by showing a few screenshots. Being a multi-user cloud-based application, AIUTO can be used by a distributed SW testing team. AIUTO has a graphical web interface to allow SW testing engineers to access into their own workspace. AIUTO allows designing and implementing test cases and monitoring Key Performance Indicators (KPI) in a single configurable dashboard (see Fig. 3). Based on access privileges and specific configuration, SW testing engineers can cooperate to e.g. balance their work load or work on the same test cases in a reliable way.

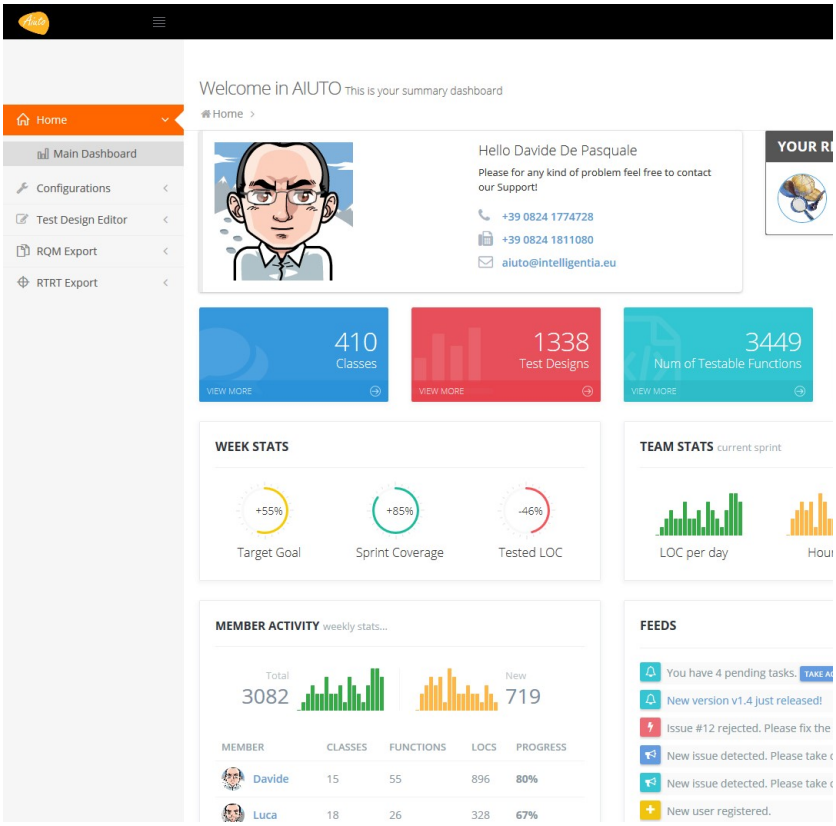

Figure 4 AIUTO user dashboard

After having logged into AIUTO, a SW testing engineer can be notified by the fact that a new version of his/her assigned SW modules are available. AIUTO synchronises with the OBSW repository change logs and provides the list of the added/updated OBSW source files. AIUTO is able to process the OBSW structure in terms of packages, classes and methods. This way, the SW testing engineer can evaluate the effort in testing the updated/added OBSW source files and organise the workload accordingly. Only the necessary information needed for testing is presented via the web editor, that is to say, the list of the methods to be (re-)tested, see Fig. 4.

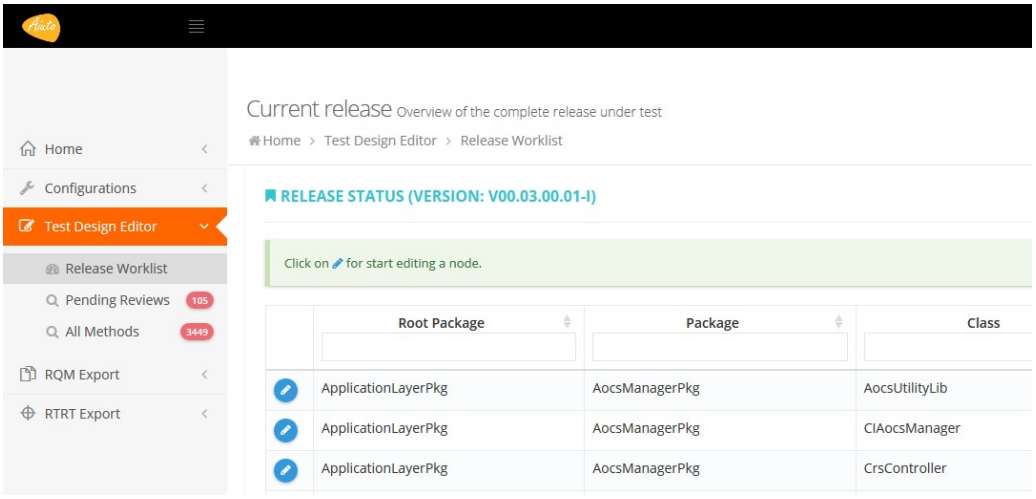

Figure 5 OBSW repository check and workload setup in AIUTO

Such web editor also facilitates the test case specification for each C function under test. In particular, starting from the admissible values of the input and output parameters from the SW Detailed Design, it supports the creation of the Equivalence Classes. From such Equivalent Classes, AIUTO can automatically derive all the possible tests case

variants. Moreover, based on some filters defined by the user (e.g., specific not significant test data variants), AIUTO automatically can prune the set of the test cases, see Fig. 6.

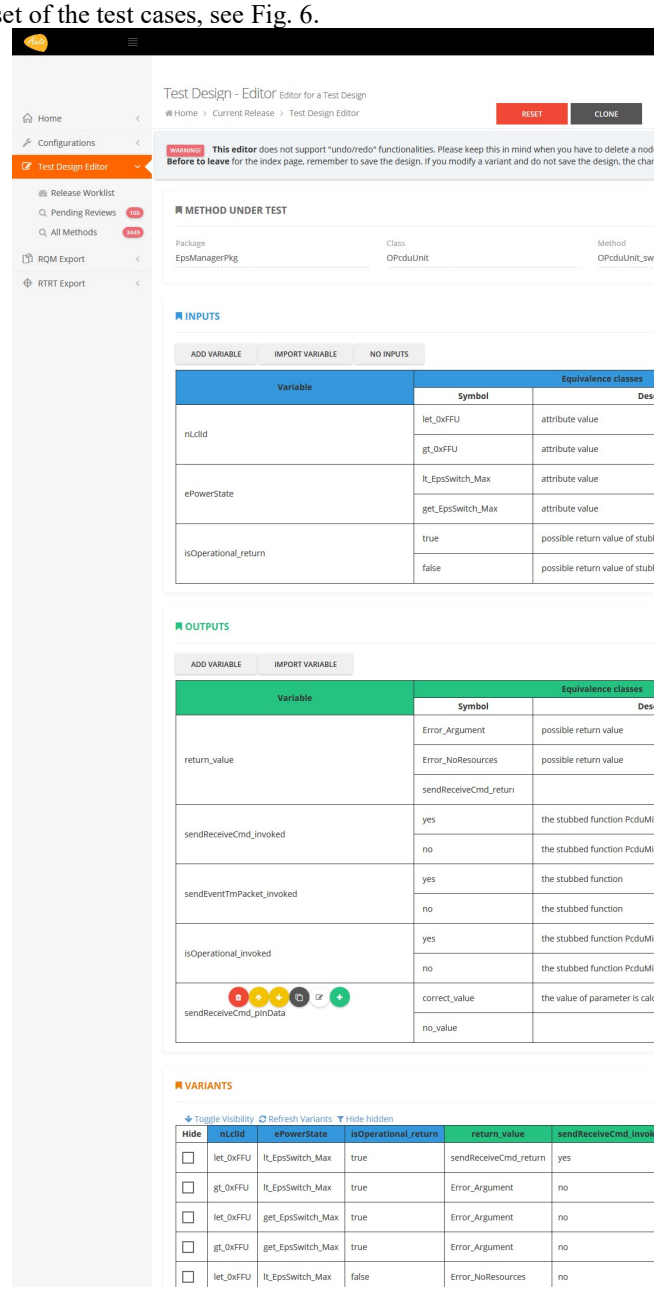

Figure 6 AIUTO test case specification editor

Once the test cases have been designed, they can be implemented in the target scripts by using an on-line code editor with grammar checks. Different scripting languages can be used e.g., IBM RTRT, Cantata, third-party testing tools. Once the test case implementation is complete, AIUTO can plan and trigger the execution of the test campaign via the backend testing software tools by using the licences available and by managing parallel and multi-node execution. Finally, AIUTO HTML artefacts can be used to streamline the production of SW testing documents e.g., the SW test case specification and the SW test procedure.

AIUTO has been routinely used by Intelligentia for the OBSW Unit tests in Meteosat Third Generation satellite [9]. The OBSW testing team has experienced an increase in terms of performance (e.g., reduction of the time spent for specifying unit test cases) and number of identified SPRs.

CONCLUSION & FUTURE WORK

This paper has presented AIUTO (Aerospace Integration and Unit Test Organizer), an integrated multi-user cloud environment to automate the SW unit test & integration testing process for embedded systems even in presence of geographically-distributed teams. Specifically, AIUTO has been used to support the design, implementation, execution of SW unit & integration tests for spacecraft On-Board SW (OBSW) and aids the related test report analysis.

Currently in AIUTO the equivalence classes shall be defined manually. It is planned to automatize this task for offering a sort of auto-completion capability of the test design for not-complex functions. It will be integrated with a static code analysis for generating symbolic tables in order to allow the testers to access detailed and structured information about macro, constants, structs, unions, and typedef definitions in the source code. In addition, using the result of the static analysis performed on the code, some functions and the skeleton of the test case design can be proposed as a draft (e.g. an if statement involving a constant can be automatically modelled as an input and output in the test design and the variants will be auto-generated accordingly). On the execution side, AIUTO will implement a fair scheduling algorithm for deferred test execution based on SVF status, availability of SDVE licences, and complexity of the test (avoid a FIFO approach for huge workload that can take hours blocking a testing node). Options to integrate with Vector Cast and a better integration with Cantata++ will be explored.

The AIUTO marketplace is “ready to sell” and Intelligentia is evaluating to grant access only to ISTQB (International Software Testing Qualifications Board) certified personnel all around the world. But this can be configurable also at the project level. For example, we can also have the possibility to use AIUTO for internal projects, thus without any involvement from external testers that are not invited by the project owner. This way, AIUTO can be adopted for both restricted and commercial projects. The last aspect (at the moment at low priority at Intelligentia) is the possibility of installing the overall AIUTO application at customer premises on dedicated hardware or virtual environment (e.g. support to military projects).

REFERENCES

[1] C. Ebert and C. Jones, "Embedded Software: Facts, Figures, and Future," in *Computer*, vol. 42, no. 4, pp. 42-52, April 2009.

[2] I. Sommerville, An Integrated Approach to Dependability Requirements Engineering. *Current Issues in Safety-Critical Systems*, Springer London, pp. 3-15, 2013.

[3] J. S. Norris, "Mission-critical development with open source software: lessons learned," in *IEEE Software*, vol. 21, no. 1, pp. 42-49, Jan.-Feb. 2004.

[4] *The CHAOS Manifesto: Think Big, Act Small*. The Standish Group International (2013).

[5] P. Duvall, S. M. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.

[6] I. Fernandez, A. Di Cerbo, E. Dehnhardt, and M. Tipaldi, “Test Automation for Critical Space Software,*” 2016 IEEE Metrology for Aerospace (MetroAeroSpace)*, pp. 551-555, 2016.

[7] *ECSS-E-ST-40C: Space Engineering – Software*. European Cooperation for Space Standardization, 2009.

[8] *ECSS-Q-ST-80C: Space Product Assurance – Software Product Assurance*. European Cooperation for Space Standardization, 2009.

[9] M. Tipaldi, C. Legendre, O. Koopman, M. Ferraguto, R. Wenker, and G. D’Angelo, “Development strategies for the satellite flight software on-board Meteosat Third Generation,” *Acta Astronautica*, vol. 145, pp. 482-491, 2018.