DOI 10.5162/ettc2020/6.6

Automated closed-loop testing of cyber-physical systems

Gonçalo Valadas (Principal Engineer), Nuno Bustorff Silva (Business Development Manager)

Critical Software S.A., Pq. Ind. de Taveiro, Lt 49, PT-3045-504 Coimbra, Portugal
goncalo.valadas@criticalsoftware.com, nuno.bustorff@criticalsoftware.com

Abstract: Modern systems envisages fusion of technologies across the physical, digital and biological worlds, an incoming era of cyber-physical systems powered by artificial intelligence (AI), the Internet of Things (IoT), autonomous vehicles and big data. More than any other industry, Aerospace & Defence innovation and production demand precision and high-quality products. Aerospace & Defence Industry must adopt features of modern product development like Agile methodologies, DevOps, and tests automatization, to accelerate time to market and business value.

The increasing complexity and validation costs raises an unaffordable trend, particularly in the evolution from hardware-based systems to modern software-based systems, where the costs to develop, integrate, and maintain software continues to grow at an unpredictable rate.

With the objective of providing an infrastructure for testing high integrity cyber-physical systems in closed-loop, Critical Software has designed and developed an end-to-end process, integrating Agile methodologies, client toolchain and an automation testing platform. This integrated solution enables real-time control and joint and flexible integration of simulation models and physical components or subsystems to build a closed-loop test environment adaptable to any development environment.

This paper presents Critical Software's System Validation approach to reduce both the time and cost of testing high-integrity and complex systems for the Aerospace & Defence Industry.

Keywords: Defence, Aerospace, Agile, DevOps, Testing Platform, Closed-Loop Testing, Regression Testing, Security Testing, Cyber-Physical Systems, Complex Systems, Validation

1. Introduction

Developments in Artificial Intelligence (AI), the Internet of Things (IoT), autonomous vehicles and big data enabled the creation of new aerospace and defence systems that historically were mainly hardware-based but are gradually becoming more software-defined [1]. Software defined means that items or functions that were mainly physical have now become virtual, controlled and automated by software [2]. These systems integrating computation, networking and physical processes are called cyber-physical systems. Examples of this systems are smart measuring systems, robotics, wireless sensor networks among others.

This evolution was triggered by the need of OEMs to have faster access to the market, boost efficiency and produce economy of scale. The usage of virtualization, cloud computing and artificial intelligence made possible new technology developments and complex networks generating huge amounts of data to be analysed and measured [3] [4].

At the same time, the effort and time needed to validate these systems also grow at an exponential rate because of the increasing configurability of these systems, the number of external interfaces, and the need for safety-related certification.

Within this scenario of constant evolution and adaptation, it is impossible to continue to execute these developments in a classical waterfall lifecycle. Verification and Validation activities must be performed from the beginning of development and an iterative approach that enables the system to be developed and validated incrementally is required. The new Agile approach seems to solve all these problems, but a formal process must be defined to guarantee that the System Under Test is able to accomplish not only the functional requirements but, sometimes even

more importantly, meets all non-functional requirements when complete [5].

Validation activities shall be performed over and over again, regressively, validating disruptions, so an automated approach is at once possible and necessary. Although it is understandable and easier when it comes to test software, it becomes more complex when the testing environment also incorporates hardware.

Beside the automation, another important thing is to have all these activities integrated into the customer's build and deploy environment. Establishing a continuous integration environment improves reliability and robustness of the validation process, therefore also improving the reliability and robustness of the system under validation as well as its overall quality.

To overcome these challenges, Critical Software defined a process to validate high integrity systems, addressing the validation of software and hardware-in-the-loop and resulting in important cost and time savings, as presented in the following sections.

2. Integrated Validation Process

Critical Software's integrated validation process enables the specification, implementation and execution of the necessary test cases to validate the System Under Test. It is composed of four main activities as outlined in the figure below.

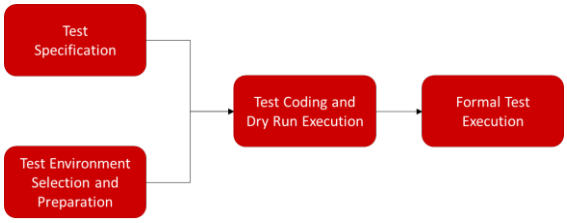

Figure 1: Process for system validation

2.1 Test Specification

In order to properly establish a test specification, the first step should be to gain knowledge about the system. This is done through performing interviews with the development team, reading the system requirements specification (SSS), and reviewing the existing interface control documents (ICDs).

The validation and integration strategies are defined at this stage. The testing methods and activities to be performed are identified, including the different types of tests that the validation process will have to execute (e.g., security, performance, usability, other functional or non-functional tests). The criticality of requirements and system components, software or hardware, are thoroughly detailed. Certification against safety standards, e.g. IEC 61508 or its industry specific derivatives, require that these activities are documented to provide evidence to the certification authority.

Then, the test plan to validate the system is created, which covers all these definitions. Because the test outputs will be validated against expected values, in this phase the applicable thresholds used in resulting assessment of non-deterministic systems are also defined, as these systems are not usually precise but will fit into a range of acceptable outputs.

Mapping to the Agile world, this could be the Product Vision and Backlog creation phase. The test sets are the epics and the test cases are the user stories described with the acceptance criteria.

2.2 Test Environment Selection and Preparation

The environment selection and preparation are important in establishing the basis for test automation. The identification of the target system, both physical and logical interfaces, are the most relevant aspect in environment preparation.

To perform the automation of integration and system tests, Critical Software have developed a System Validation Facility platform, an asset providing the basis for the development of automated test sets that will exercise the system and capture their outputs to validate them against expected values.

The system validation facility is an automated execution environment responsible for test execution, including the simulation of the necessary input conditions and the evaluation of response values. It also compares expected and actual test case output values, validating

them according to the acceptance criteria defined for that test case [6].

Another important feature of the system validation facility is the ability to perform fault injection at hardware level. The ability to inject faults at hardware level enables the tester to simulate internal faults on the System Under Test and evaluate the reaction to errors inserted in different parts of the system, such as the processor registers, the memory, or the application code. Faults are injected with minimum interference with the target system workload, making it almost non-intrusive. This enables a very robust test capability of the safety and reliability mechanisms which are not easily testable.

In the Agile process this phase matches sprint 0, where the implementation sprints are prepared, and the backlog is prioritised.

The environment preparation also includes the setup of a continuous testing environment where automatic build, test and static analysis is performed when a change occurs in the version control system repository. At this point, three major verifications are being performed: (a) that the build system is correctly generating the target system; (b) the target system passes all defined test suites; and (c) the system code complies with the defined code quality rule set.

2.3 Test Coding and Dry Run

Test coding activity is where test cases defined in the specification are implemented. This is a manual coding activity where developers write the code that executes steps defined in the test case. The usage of Artificial Intelligence techniques in this process to automatically generate test case code would be beneficial as systems become more complex. Input vectors that stimulate the implemented test cases will be defined separately in order to change these inputs without having to rebuild the test cases in question.

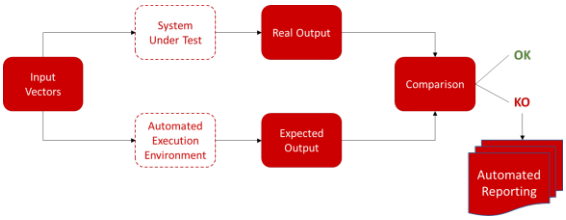

Figure 2: Test Execution Scenario

The first item to validate is the correct implementation of system interfaces. These will transport the inputs of the test cases to the System Under Test and return the relevant outputs.

During this activity, dry runs of the test suites shall be performed to guarantee that tests execute as specified and, if not, are able to debug and correct the test. At this time, the first issues are raised allowing developers to fix arising problems before the formal test runs.

The usage of an Agile methodology to implement the tests and verify their correctness enables incremental validation of the test implementation and the System Under Test, identifying the first issues to be raised during these first dry runs.

Due to automation, these test suites can be run when required to perform *ad hoc* test runs or execute regression testing integrated into the continuous integration environment.

When the test coding is finished and all tests are behaving as specified, the automation testing environment creation process has been completed.

2.4 Formal Test Execution

The formal test runs are executed according to plan, running the released tests against the released system.

These activities are fully automatic and consist in the execution of all tests which are to be formally used for the certification record. They automatically produce a report with the validation evidence at the end of the process.

When the entire system is finished, a final formal test is executed to produce the documentation used for certification purposes. At this point, the development must be frozen and the version to be formally tested should be released.

After this, the System Under Test is ready for certification, but the automated environment created can be continuously used to support future enhancements of the system. If those enhancements change the expected behaviours or outputs, the automated execution environment must be adapted accordingly to support the new behaviour.

3. Conclusion

This paper presented an integrated process allowing OEMs to enhance their product development by automating the validation process and to reduce the total cost of ownership of critical systems through the automation of several Verification and Validation activities.

The usage of the System Validation Facility enables the automation of the tests using hardware-in-the-loop. The automation of test suites reduces the time and effort taken to realise test campaigns and eases the process of producing certification records and validation evidences. The ability to perform regression testing gives engineering teams the time to focus on high value tasks instead of spending time re-running test suites manually.

Fault injection at hardware level enables the tester to simulate internal faults on the System Under Test and enable the validation of hardware built-in tests as well as the reaction to errors inserted in different parts of the system such as the processor registers, the memory, or the application code.

The usage of a SCRUM approach in developing the automated execution environment is beneficial as it means change can be managed as an integral part of the process and also shortens feedback loop.

In future work, the Critical Software's System Validation Facility aims to use Artificial Intelligence to generate test cases that will simplify the test specification activity and use Artificial Intelligence to develop test data for use as input vectors in test coding and dry run activities.

Based on the experience of some of our clients, this process reduced the TCO of critical systems by up to 50%.

References

[1] Grant J. McDonald, Yvon Audette (2018), 'Making the move to Industry 4.0', published in Canadian Defence Review Magazine, February 2018.

[2] Sadiku, M & Nelatury, Sudarshan & Musa, S. (2017). 'Software defined everything', 4. 48-50.

[3] Daliri, A, Das, R, Orifici, A, Marzocca, P and Cole, I (2019), 'Virtual Design, Optimisation and Testing (VDOT) framework for innovative sustainment', in Proceedings of the 18th Australian International Aerospace Congress (AIAC18: 2019), Melbourne, Australia, 24-26 February 2019, pp. 122-128.

[4] Kim, Beom-Su & Kim, Ki-Il & Shah, Babar & Chow, Francis & Kim, Kyong Hoon. (2019). 'Wireless Sensor Networks for Big Data Systems'. Sensors. 19. 1565. 10.3390/s19071565.

[5] Valadas, G. (2019), 'White Paper - Using Agile to Develop High Integrity Systems' published by Critical Software, October 2019.

[6] A. Chaves, R. Maia, C. Belchior, R. Araújo and G. Gouveia (2018), 'KhronoSim: A Platform for Complex Systems Simulation and Testing' 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, 2018, pp. 131-138.