

The STADY project (Applied Static and Dynamic Verification of Critical Software) is an R&D activity in response to the Announcement of Opportunity "Call for Innovative Technology Proposals", Theme 1 - Software. The project aims at research and demonstration of an innovative technique for the verification of the safety and reliability characteristics of software in space applications. This new technique is based on the joint application of static and dynamic verification methods to critical software. The activity defines in detail the method to be used and how the Static analysis can be complemented by Dynamic verification. Demonstration and training of the applicability of the proposed innovative technique has been performed on two case studies:

- 1) the open Ravenscar kernel – ORK;
- 2) and an on-board software test application build on the OBOSS-II reuse library.

The project has performed substantial analysis on available methods and tools and classified them according to criteria's of added value. It further defines the new STADY methodology, which combines a static method, SoftWCare, which is based on SFMEA and SFTA, with a dynamic technique, Xception, based on fault-injection. The synergetic combination of both techniques brings a huge potential since each one contributes to mitigate the shortcomings of the other. Application of the SoftWCare method provides important inputs to the definition of suitable Xception-fault-models to be applied to the system under evaluation. The effectiveness of fault-injection activity is heavily depending on the definition and representativeness of the fault model.

The SoftWCare method consist of:

- SFMEA analysis, which identifies the failure modes, analyzes their effect, and identifies potential causes. This analysis is performed at functional level and it is intended to identify the critical functions and the causes of their potential failure modes. A software failure modes and effects analysis table is produced identifying all potential failure modes and their effects and causes.
- SFTA analysis, which determines the software faults causing the failure modes identified in the previous analysis step. This analysis evaluates the design and the code of the software, starting from the 'top events' identified in the SFMEA and finishing by the last variable fault potentially causing a failure. For each function, data, interface, etc, all potential software faults are evaluated by systematically check of all fault types identified in the taxonomy.  
Finally, the static analysis joint evaluation is performed to assess the results of the two analyses and to document the results in a 'static analysis reports'. This report includes the recommendations intended to eliminate the identified software faults, and provides definition of test cases suitable as input to the dynamic analysis.

The dynamic analysis is based on the use of the Xception toolset and includes the following step.

- Test preparation. This first step of the dynamic analysis is used to collect all documentation and data available of the software under evaluation for the subsequent analysis steps. Available documentation and source code is evaluated

and deeply scrutinized for the complete understanding of the software target of evaluation.

- Test case execution is based on mutation technique which generates load modules with the specific fault condition inserted. The test case generation is supported by tools that allow fast mutation generation and automatic test result evaluation.
- Finally, the dynamic analysis conclusion is performed to assess the results of the tests executed and to document the results in a 'dynamic analysis report'. This report includes the recommendations (complementing and verifying the first set of recommendations after the static analysis) to eliminate the software faults found. Additionally it performs recommendations feedback input to the static analysis, identifying additional areas that deserves further analysis and confirms/rejects the identified faults from the static analysis.

The final presentation will explain the methods used and the lessons learned from using it on the two test cases.