Automated Component Testing of Evolutionary Software: An Industry Example

Zlatko Franjcic, Morten Fjeld
t2i Lab, Chalmers University of Technology
Gothenburg, Sweden
{*zlatko.franjcic, fjeld*}*@chalmers.se*

***Abstract*—We present a summary of a problem analysis examining the challenges of introducing automated component black-box testing for an evolutionary grown software system. We outline the system and illustrate the challenges associated with testing. To explore to what extent black-box testing is feasible under a set of given conditions, we developed a prototype test automation framework. We describe the lessons learned in the process.**

***Keywords*-test automation; black-box testing; regression testing; architecture degradation; coupling; industry example**

I. Introduction and Background

We consider a commercial software product used for processing digital signals obtained from sensor measurements.[1] The product's code has grown evolutionary during more than a decade with contributions from various individuals. Maintenance cost of the system has risen over time due to software architecture degradation [1], in particular increased coupling. At the same time, black-box testingeffectiveness has been hampered by inadequate requirements and incomplete documentation of software interfaces. The combination of these problems has led to increased costs and risks when implementing changes.

II. Testing Challenges

The software manufacturer's previous test strategy foresaw manual system-level tests only. With additional customer needs emerging over time, changes to the long untouched signal-processing algorithm implementations became unavoidable. The test strategy's scope was broadened to include testing of novel algorithms in parallel with the existing implementations. Defining testable requirements for the signal-processing pipeline is not straightforward. Floating-point arithmetic, real-time constraints and an error propagation function across pipeline stages that is not fully understood amount to a multitude of objectives which are in part not adequately specified. Additionally, lacking documentation and coupling of system components further obstruct testing towards ensured confidence in the correct functioning of the software.

[1]Due to a confidentiality agreement we choose not to disclose the name or manufacturer of the software product, nor any of the software's specifics.

III. Test Automation Prototype

Our goal with this work is to assess to what extent black-box testing of system components in an automated fashion is feasible given the challenges described. We decided to develop a prototype solution to gather tangible evidence for the assessment. In particular, the prototype should calculate a set of predefined quantitative measures during test execution to support diagnosis of how a particular change to the signal-processing algorithm implementations affects a functional or performance characteristic at different stages of the signal-processing pipeline.

This resulted in a test automation prototype based on the equivalence partitioning approach, with manual specification of example inputs and acceptance criteria. The prototype allows for testing individual stages of the signal processing pipeline, including the possibility to simultaneously test alternative implementations.

IV. Lessons learned

We used the prototype to test a particular stage of the signal-processing pipeline given a set of recorded sensor measurements as input and maximum admissible deviations from target output signals as acceptance criteria. While we found the prototype to correctly identify the test cases failing the criteria, the quantitative measures alone were not enough to explain the causes of the deviations. We regard the current prototype to be suitable for regression testing with binary outcome (pass/fail). In its present form, however, it is not generic enough to be used for testing of arbitrary software systems. More informative test results would require adding support for more analytical testing methods [2].

Acknowledgment

Work supported by the EU FP7 People Programme (Marie Curie Actions) under REA 289404 and 290227.

References

[1] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: understanding causes and a qualitative model," in *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*. IEEE, 2014, pp. 85–92.

[2] R. A. Santelices, Y. Zhang, H. Cai, and S. Jiang, "Change-effects analysis for evolving software." *Advances in Computers*, vol. 93, pp. 227–285, 2014.