Poster: Results from Multi-faceted Software Reliability Assessment Studies*

Yaping Luo
Altran Netherlands B.V.
Netherlands
yaping.luo@altran.com

Egbert Touw
Altran Netherlands B.V.
Netherlands
egbert.touw@altran.com

ABSTRACT

Software quality aspects and in particular software reliability is hard to measure. Current tools or techniques are not sufficient to support us to get insights in the software reliability risk in our customer's products. In this poster, we present a multi-faceted software reliability approach. Besides an industrial case study is also described with four assessment studies. The results of the case study have been well accepted by our customer and by the supplier of this customer. Moreover, our approach can be applied in the automotive domain as it is and even be improved by exchanging some of the process assessment components by the parts from automotive reference models, such as A-SPICE[1].

KEYWORDS

Software Reliability; CMMi; TMMi; ISO25010; Quality Improvement

1 INTRODUCTION

Based on fault Introduction and finding theory (Figure 1) at least 5% of faults are never found, which means there are potential software reliability risks. However, software quality aspects and in particular software reliability is hard to measure. Although software reliability growth models (SRGMs) [1] have been developed to estimate or simulate software reliability measures such as the number of remaining faults, software failure rate, and software reliability. To the best of our knowledge, there are no existing tools or techniques that are able to give our customers insight in the software reliability risks and status of their products. To get insights of different quality aspects of software, we propose a multi-faceted approach for software reliability assessment.

* Produces the permission block, and copyright information

[1] Automotive process standard

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195068

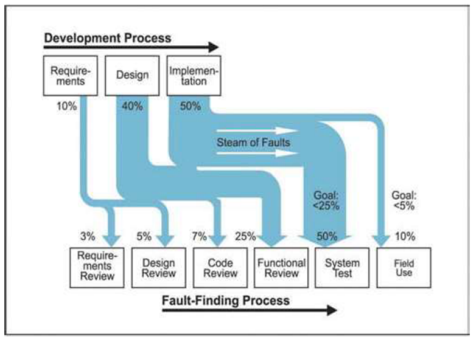

Figure 1. Fault Introduction and Fault Finding Theory

2 MULTI-FACETED SOFTWARE RELIABILITY ASSESSMENT

A multifaceted assessment technique [2] is presented to identify the practices in the product lifecycle processes that might introduce not-found faults in software. The software reliability assessment consists of two parts: process assessment and code quality assessment. Our multi-faceted software reliability assessment is based on selecting process areas to be assessed from proven assessment methodologies (CMMi-DEV [3], CMMi-SRV [4], TMMi [5]), and combine the results from this process assessment with the results of an ISO25010 based code quality assessment.

2.1 PROCESS ASSESSMENT

For process assessment, the CMMi for development and CMMi for service are used as reference assessment models. A number of process areas are selected from these two standards. Then each practice in the selected area is scored according to the software reliability assessment code (Figure 2).

Within the reliability and risk analysis process, the probability that an assessed practice introduces a not-found defect that affects reliability is classified according a combination of "Consequence" (the relative expected frequency of the assessed practice in which a not-found defect can possibly cause a failure in the product or an

Software Reliability Assessment Code

Consequence / Satisfaction	Insignificant	Minor	Moderate	Major	Extreme
Not Satisfied	3	2	1	1	1
Partially Satisfied	4	3	2	1	1
Largely Satisfied	4	3	3	2	1
Fully Satisfied	4	4	4	4	4

Figure 2. Software Reliability Assessment Code

applied service) and the "Satisfaction" level of the assessed practice (the relative likelihood that a not-found defect will be introduced and will cause a reliability issue somewhere in the future). For example, when code coverage is not measured (satisfaction = partially) there is a significant chance that not all code is tested and that not all defects are found. Reliability classification for code coverage will be "1". When all software code is covered by tests (satisfaction=Fully) the chance is low that defects are not found. Reliability classification for code coverage will be "4".

This reliability classification makes the assessment result quantifiable (a reliability number can be calculated based on the assessment result) that make development and service organizations comparable to each other with respect to software reliability capability.

2.2 CODE QUALITY ASSESSMENT

Code quality assessment is carried out against ISO25010. The following subjects are investigated: Method size, Cyclomatic Complexity, Fan in/out, Cyclic Dependencies, Dead code, Coding rules, Code smells. Duplicated Code has been identified with the tool "CopyPasteDetector" (CPD) from the PMD-suite of Sourceforge. Besides, the code quality assessment also covers Archive structure; Product variants; Reliability mechanisms and Engineering practices. Figure 3 gives detailed requirements on these subjects, which are used during the code quality assessment.

Topic	Judgement / Criteria
• Code size;	Percentage testcode ~40%, comment to code ration ~ 50%
• Function size;	Functions >100 lines are candidate for refactoring
• Complexity;	Calculated complexity <20
• Cyclic Dependencies;	Count cyclic dependencies; target = 0
• Code duplication;	Target < 10%
• Dead code;	Target < 0%
• Coding rules;	For C/C++ MISRA2012, target = no critical rule violations
• Code smells;	Sniff on a selection of code files by expert
• Archive structure;	Check on architecture rules and clean structure and ease of navigating
• Product variants;	Configuration management
• Reliability mechanisms;	E.g. error handling, code reviews, exception handling
• Engineering Practices	Coding, reviewing, problem solving, commenting, check in/out, unit testing

Figure 3. Detailed Requirements on Code Assessment Subjects.

3 RESULTS

Both customer and suppliers were surprised by found software reliability risks. Software reliability was not specified for their products. Suppliers did not design for or measure software reliability aspects. Figure 4 shows a summary of the process assessment results at four suppliers.

Based on the assessment approach, a number of improvement points were given by us to the customers to address the potential reliability risks: 1) Include software reliability requirements in customer specification. 2) Set requirements for software quality management of suppliers. 3) Identify KPI's for software development and test processes. 4) Define a software quality assurance process and role. 5) Define non-functional requirements for software. 6) Demand a software FMEA where software failure modes are identified, 7) Apply tools to identify software reliability risk in code-base. 8) Carry out re-assessment after some time.

Process Area	Supplier 4	Supplier 3	Supplier 2	Supplier 1
Requirements Management	3,76	3,71	3,41	3,65
Process & Product Quality Assurance	3,38	3,31	3,00	1,19
Configuration Management	3,79	3,74	3,26	3,68
Service Delivery	3,80	3,90	3,90	3,30
Incident Resolution & Prevention	4,00	3,78	4,00	3,48
Service System Development	4,00	4,00	3,83	3,63
Service Continuity	4,00	4,00	3,55	2,55
Requirements Development	3,82	3,77	3,41	3,64
Technical Solution	3,33	3,76	3,43	3,67
Product Integration	3,90	3,86	3,76	3,48
Verification	3,50	3,60	2,95	3,55
Validation	4,00	3,82	3,76	3,82
Total Reliability Score	3,77	3,77	3,52	3,30

Figure 4. Results of Process Assessment at Four Suppliers.

4 CONCLUSIONS

Software reliability is very important for complex systems. In this paper we applied a multiple-faceted assessment technique that is based on existing reference assessment models and techniques to get better insight in software reliability. To demonstrate our techniques, a case study has been discussed. The results of the case study have been well accepted by our customer and by the supplier of this customer.

ACKNOWLEDGMENTS

We would like to give a special gratitude to the assessment team (Mr. Robbert Schuitema (Software Architect), Mr. Rob Posthumus (Test Expert), and Mr. Jeroen Schutte (Project Manager)) who developed and executed the approach with us.

REFERENCES

[1] Xuemei Zhang Xiaolin Teng, Hoang Pham, Considering fault removal efficiency in software reliability assessment, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans (Volume: 33, Issue: 1, Jan. 2003)
[2] Egbert Touw; Multi-faceted reliability assessment techniques : An Industrial Case Study. WASA 2017.
[3] CMMi-DEV CMMi for Development Version 1.3; CMMi Product team; Improving processes for developing better products and services; November 2010; CMU/SEI-2010-TR-033
[4] CMMi-SVC CMMi for Services Version 1.3 CMMi Product team; Improving processes for providing better services; November 2010; CMU/SEI-2010-TR-034.
[5] TMMi Test Maturity Integration (TMMi) Release 1.0, Erik van Veenendaal; 2012