2013 IEEE 37th Annual Computer Software and Applications Conference

SQAF-DS: A Software Quality Assessment Framework for Dependable Systems

Junbeom Yoo , Sanghyun Yoon
Department of Computer Science and Engineering
Konkuk University
Seoul, 143-701, Republic of Korea
{jbyoo, pctkdgus}@konkuk.ac.kr

***Abstract*—This paper proposes a software quality assessment framework for dependable systems (SQAF-DS), providing a systematic way to assess software quality through test cases, indirectly. SQAF-DS intends to reduce the time and cost for dependability assessment thorough using test cases as a means of the assessment. Test cases are developed in the process of software development and used to test target system, while dependability requirements are derived from dependability analysis, such as FTA (Fault Tree Analysis). SQAF-DS formally checks inclusion relation between dependability requirements and test cases. If the formal checking succeeds, then we can assure that the dependability requirements are well implemented in the software system.**

***Keywords*-software quality; dependability; quality assessment; dependable system; test cases; formal checking**

I. Introduction

Dependability is an important prerequisite for dependable systems [1], and should be demonstrated sufficiently to get operating permissions by regulation authorities. It reflects the extent of the user's confidence that it will operate as they expect and that it will not *fail* in normal use [2]. While many researchers have defined the dependability in slightly different ways [2]–[4], we adopt the one of Sommerville, a tuple of four principal dimensions such as *safety*, *security*, *reliability* and *availability*. Each dimension includes analysis techniques for achieving itself and assessment methods/measures.

Dependability assessment is an important activity as well as dependability achievement (i.e., analysis) itself, since the former plays a role in deciding when to stop the effort for the latter. Analysis techniques take much time and cost, and a prompt decision whether to keep them up while preserving a required level of dependability is one of the key factors to cost-effective software development. This paper proposes a way to reduce the effort for dependability assessment; Software Quality Assessment Framework for Dependable Systems (**SQAF-DS**). SQAF-DS intends to reduce the time and cost thorough using test cases as a means of the assessment. Test cases are developed in the process of software development and used to test target system, while dependability requirements are derived from dependability analysis, such as FTA. SQAF-DS checks formally inclusion relation between dependability requirements and test cases. If the formal checking succeeds, then we can assure that the dependability requirements are well implemented in the software system, indirectly. SQAF-DS also provides a systemic way for transforming dependability requirements and test cases into inputs to formal checking techniques such as the VIS equivalence checking [5] and the SMV symbolic model checking [6].

SQAF-DS needs no additional cost for developing test cases, since they are developed in the process of software development. On the other hand, SQAF-DS should provide a systematic technique for eliciting requirements from the results of dependability analysis and then transforming them into specific forms of properties for formal checking purposes. Test cases also should be transformed into specific input forms to formal checking methods. The transformation and formal checking techniques adopted would vary depending on the forms of dependability requirements elicited and test cases. The following section introduces the details on SQAF-DS.

II. SQAF-DS

⟨Fig.1⟩ overviews the proposed software quality assessment framework for dependable systems - SQAF-DS. It intends to reduce the cost of software quality assessment thorough using test cases as a means of the assessment. It uses formal checking techniques to ascertain inclusion relation between dependability requirements and test cases. Dependability requirements and test cases are transformed into specific inputs to formal checking methods. If the formal checking produces `TRUE`, then we can conclude that the dependability properties are well implemented into the software system and we do not need any other dependability analysis to assess the properties any more. On the other hand, in case of `FALSE`, we cannot check the properties through the test cases and have to use other assessing methods, the same as before.

A. Formal Checking

The formal checking depicted in the middle of ⟨Fig.1⟩ plays an important role in SQAF-DS. It checks inclusion or equivalence relationship between dependability properties and test cases. If the checking produces `TRUE`, we can confirm that the dependability requirements are includes in (or equivalent with) the test cases. If the test cases are all passed successfully, whether unit tests or system tests, we can assure that "*The dependability requirements are well implemented in the target software system*". If the checking results in `FALSE`, we cannot

0730-3157/13 $26.00 © 2013 IEEE
DOI 10.1109/COMPSAC.2013.118

be sure "*The dependability requirements are NOT implemented yet.*", since they may be implemented in other ways which cannot be assured by the test cases.

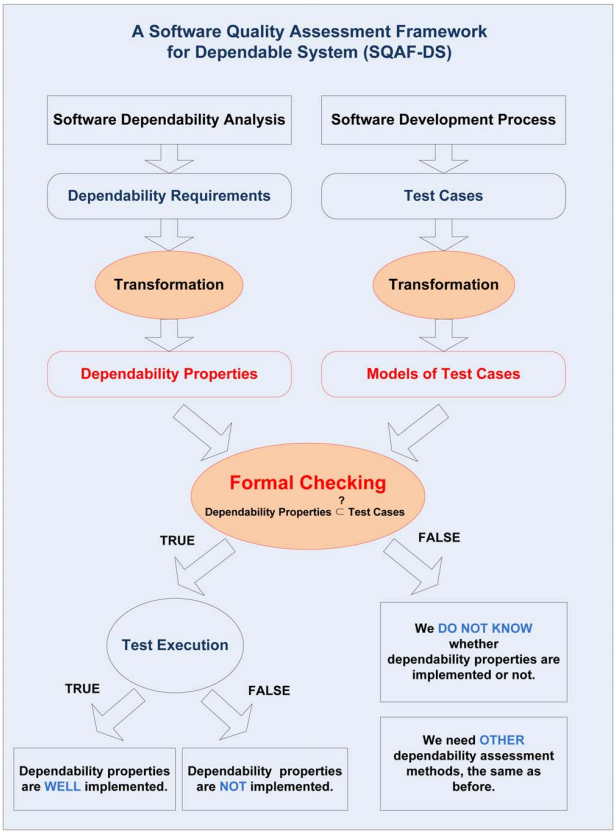

Fig. 1. A software quality assessment framework for dependable systems (SQAF-DS)

Model checking [7], equivalence checking [8] and set inclusion [9] are candidates for the formal checking methods. In case of model checking, we typically need two inputs; temporal logic properties [10] and a model of finite states machines, which depend on the model checker we decide to use.

B. Dependability Requirements

Dependability requirements are derived from the results of dependability analysis. For example, in safety dimension, dependability requirements can be elicited from '*minimal cut-sets*' [11] of FTA (Fault Tree Analysis). In case of security, '*security patterns*' would be useful as candidates for dependability requirements. Transformation from dependability requirements into specific forms of properties or programs depends on the formal checking methods which we use. If we use the VIS verification system for behavioral equivalence checking, we need to transform the dependability requirements into a Verilog program. In case of symbolic model checking such as the SMV model checker, they should be transformed into CTL properties. If we use SAT/SMT solvers such as Yices and MiniSAT, we have to transform the dependability requirements into specific forms of propositional logics.

C. Test Cases

The other input to formal checking methods is test cases. Test cases are developed in the process of software development and derived from requirement/design specifications (for functional test) or source code (for structural test) by development teams. SQAF-DS transforms test cases into specific input programs of formal checking methods as dependability requirements, too. If formal or semi-formal specifications are used, then the transformation would be mechanized.

III. Conclusion and Future Work

This paper proposes a software quality assessment framework for dependable systems - ***SQAF-DS***. It intends to reduce the cost for assessing the software quality (*i.e.,* dependability) through using test cases as a means of the assessment. We are now planning to perform a full-scale safety assessment on system-level dependability requirements and test cases of a control software of nuclear reactor protection system in Korea as well as the preliminary examples of [12].

Acknowledgment

This research was supported by the MKE, Korea, under the Development of Performance Improvement Technology for Engineering Tool of Safety PLC program supervised by the KETEP" (KETEP-2010-T1001-01038) and a grant from the Korea Ministry of Strategy, under the development of the integrated framework of I&C conformity assessment, sustainable monitoring, and emergency response for nuclear facilities.

References

[1] D. Jackson, M. Thomas, and L. I. Millett, *Software for Dependable Systems: Sufficient Evidence?*, N. R. C. Committee on Certifiably Dependable Software Systems, Ed. The National Academy Press, 2007.
[2] I. Sommerville, *SOFTWARE ENGINEERING (8th)*. Pearson College Div, 2006.
[3] J. C. Knight, "Safety critical systems: Challenges and directions," in *International Conference of Computer Safety, Reliability and Security*, 2002, keynote.
[4] R. S. Pressman, *SOFTWARE ENGINEERING (6th)*. Mc Graw Hill, 2005.
[5] R. K. Brayton and et. al., "VIS : A system for verification and synthesis," in *the Eighth International Conference on Computer Aided Verification, CAV '96*, 1996, pp. 428–432.
[6] K. L. McMillan, *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Programming Languages and Systems*, vol. 8, no. 2, pp. 244–263, 1986.
[8] S.-Y. Huang and K.-T. Cheng, *Fromal Equivalence Checking and Debugging*. Kliwer Academic Publishers, 1998, ch. 4.
[9] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman, *Foundation of Artificial Intelligence*, ser. Handbook of Knowledge Representation. Elsevier, 2008, vol. 3, ch. 2, pp. 89–134.
[10] M. Huth and M. Ryan, *Logic in Computer Science*, 2nd ed. Cambridge, 2004.
[11] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, "Fault tree handbook, technical report NUREG-0492," U.S. Nuclear Regulatory Commission, 1981.
[12] S. Yoon, "Safety assessment of dependable software system using test cases," Master's thesis, Konkuk University, 2012.