Durability Challenges in Software Engineering

Rajeev Kumar
Suhel Ahmad Khan
Raees Ahmad Khan

Abstract: Practices show that software quality is not as high as it could be. Development organizations spend a relatively large amount of money and effort on fixing quality issues during late-stage development of software. One of the software qualities that has received significant attention in recent years is durable serviceability. Software with poor durability is likely to fail in a highly competitive market; therefore, software development organizations are paying more attention to ensuring the durability of their software. To be able to develop durable software cost-effectively, developers must investigate the connection between durability characteristics and software. In software engineering, durability is determined mainly by four characteristics; trustworthiness, human trust, dependability, and usability. To address the relationships among these characteristics, software designers analyze the durability requirements that may need to be implemented to fulfill these specific requirements of software serviceability. The main objective of this article is to gain an in-depth understanding of the relationship between durability characteristics and software.

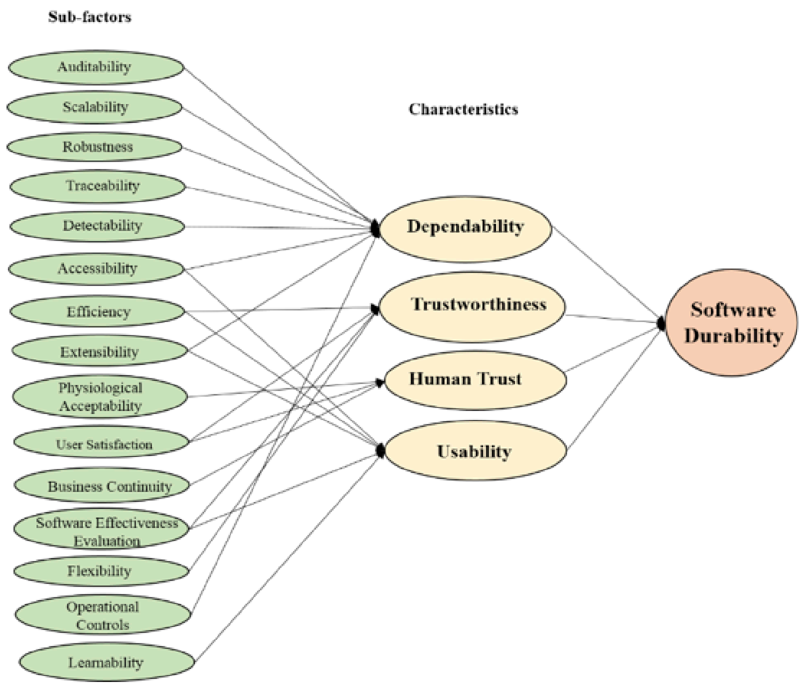

Figure 1: A structure of software durability factors [5]

1. Introduction

The software development life cycle contains many phases, including requirements engineering, design, coding, testing and debugging, and maintenance. Maintenance is regarded as the last stage of development [1]. But what if this phase continues for years? This is where the issue of serviceability comes in. Serviceability of software can be defined as the condition in which software is still useful or maintainable. Serviceability of software should be durable to achieve maintainability. Durability, in terms of software, is the time period during which software gives services.

There has been lot of work done in the field of software maintenance with regard to durability. In his article "When good software goes bad: the surprising durability of an ephemeral technology," Nathan Ensenger discusses problems in maintenance [2]. He also stated that there is a need to focus more on problems related to maintenance achievement. He stated that software durability is related to software serviceability, and it has been pointed out that achieving durability may enhance software serviceability. Service-oriented durable design of software is the aim of this study. The remainder of the paper is organized as follows: in section two, basic concepts of durability are defined. In section three, emergence of software durability is defined. In section four, successful strategies for developers are given. The paper's conclusions are contained in section five.

2. Basic Concepts of Durability

The evolving flexible environment of the early 21st century creates new challenges for all, including software developers [3]. Many programmers collaborate on each software project, with each programmer working on an individual software function or architectural component [4]. These components are often developed separately within a fixed time frame. When the time comes to bring it all together, a project manager integrates the different components as a required unit to achieve the desired result. This process makes software development a complex activity.

The complexity of software development leads to many problems, with design vulnerabilities being one of the most significant [6]. Some of the fundamental principles of design and its related systematic tools include availability, reliability, security and usability. Service of a software product is durable if it works efficiently and effectively to the user's satisfaction and for the expected duration. Many factors of software quality affect the serviceability of software, among them these few: trustworthiness, human trust, dependability, and usability. These factors affect durability directly, while factors like auditability, scalability, robustness, traceability, detectability, accessibility, efficiency, extensibility, physiological acceptability, user satisfaction, business continuity, learnability, effectiveness, flexibility, and operational controls affect durability indirectly [7] [8].

The meanings of these factors are different in a software scenario. Specifically, trustworthiness is assurance that software will perform as expected; human trust is a willingness to rely on the software with confidence; dependability refers to the ability to deliver service that can justifiably be trusted; and usability refers to how well software can be used by particular users to reach quantified results with effectiveness and satisfaction. The factors that affect durability directly and indirectly have positive and negative impacts on software service design as shown in Figure 1. A problem often

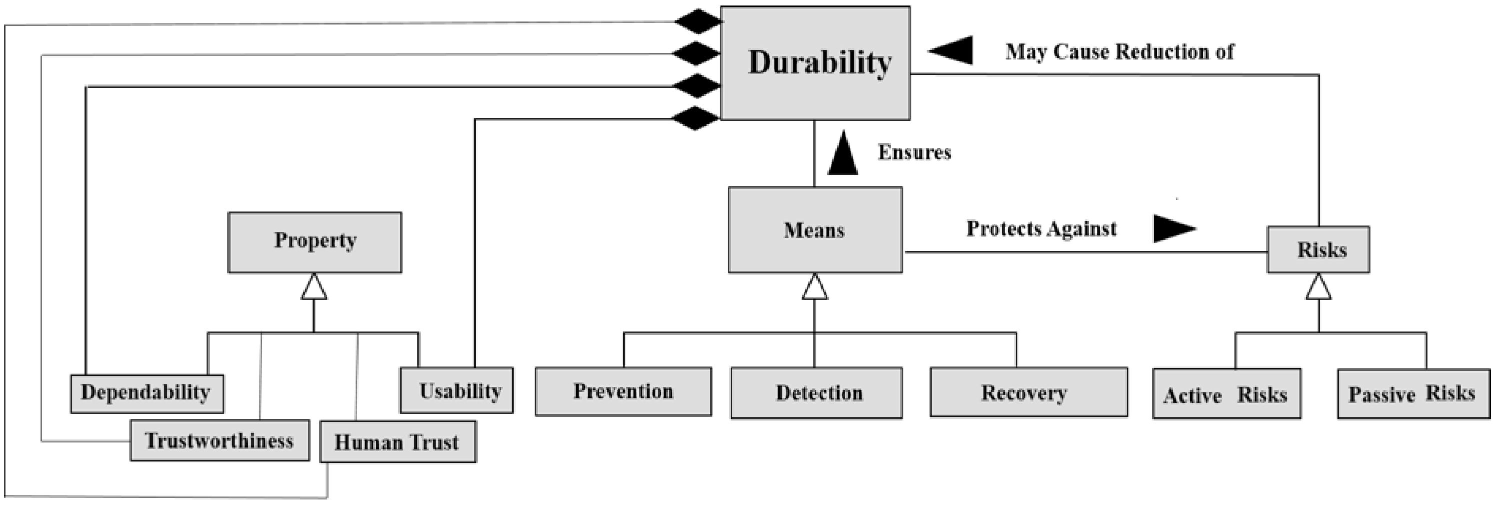

Figure 2: General activities for durability of software

comes while assuring durability by modifying the architecture of software. Modification is very expensive and time-consuming.

3. Emergence of Software Durability

Usually software is delivered without considerable security. This invites vulnerabilities. To mitigate these vulnerabilities, patching is done, which further results in more vulnerabilities in the future. It is normally expected that the design will remain serviceable for the entire life of the software and that services and qualities may come and go [9][10]. This leaves software designers and users to consider the relationship of the durability to the rest of the software architecture. Software durability is a term used to describe the usefulness and service life of a software product, which involves designing and construction with optimal maintenance [11] [12].

The term may also be used to describe the whole software development life cycle by comparing the service life of the design and its functional undesirability. A review of international research indicates that, except for operational components of software, all elements require different levels of service maintenance, repair, and replacement during the life cycle of the software development [13] [14]. The extent and strength of these services demands vary considerably, depending on how appropriately the durability of software and systems are synchronized and how accessible they are for regular maintenance, repair and replacement.

The durability of software may be expressed as a function of service quality and service life during the development cycle. There are three important service quality thresholds associated with durability: first, the quantified quality, recognized by the software developer or defined by minimum codes; second, the minimum acceptable quality, indicating the need for replacement; and third, failure. As shown in Figure 2, risk should be minimized to achieve durability of software. Two types of risk — active and passive — affect software during the development stage. Durability of software increases if risks are properly managed by means of detection, prevention and recovery.

4. Successful Strategies for Software Developers

Quality is a significant feature of software to be addressed, and durability is an important factor in evaluating software quality. Development of software design is not a one-time, built-in process; it is based on reuse of existing specifications in the market. The key point of this research work is the analysis of software's service-life relating to quality. This research is focused on increasing service life with secure and durable serviceability of software. Following are the steps which form a process that is effective in achieving durability when performed iteratively, incrementally and in parallel with the other activities, tasks and primary objectives:

—Establish durability as a powerful factor in software quality.
—Identify threat models of durability for degradation mechanisms.
—Develop a durability program plan that includes trustworthiness, human trust, usability and dependability.
—Identify and investigate their potential sources.
—Estimate the risks associated with durability for these respected assets.
—Arrange the risks according to the severity of the negative impacts.
—Identify and investigate the durable necessities of serviceability an arrangement as a benchmark for quality.
—Identify new attributes to provide a secure service-life of software for a specific duration.
—Identify durability subfactors, and determine their impact on overall software.
—Analyze software risks relating to durability.
—Make an objective to lessen the complexity of software design by establishing durability, which optimizes maintainability.
—Enhance the quality of software by improving service-oriented design.
—Calculate durability parameters using available or developed calculation models.
—If possible, update the ordinary architectural design tools for durability.

5. Conclusion

It is evident that generating a fully secure system is not possible; therefore, the creation of perfect and secure software cannot be considered the objective of evaluating software durability. Thus, the objective is to decrease the maintenance issue for longtime serviceable software. It could be concluded from the above discussion that the achievement of durable software is going to be a new challenge in the software industry. It is also as important as achieving any other attributes of quality, i.e., dependability, usability and supportability. One significant result has also been observed — that achieving durability early in development will raise the level of quality in the software. In this article, a general structure for assuring durable software is designed, and its processes are defined briefly.

Acknowledgment

This work is sponsored by UGC-MRP, New Delhi, India, under F. No. 43-391/2014 (SR).

REFERENCES

1. Kelty, C. & Erickson, S. (2015). The Durability of Software. Germany: Meson Press, 1-13.
2. Ensmenger, N. (2014). When Good Software Goes Bad: The Surprising Durability of an Ephemeral Technology. MICE (Mistakes, Ignorance, Contingency, and Error) Conference. Munich, 1-16.
3. Firesmith, D. G. (2003). Common Concepts Underlying Safety. Security and Survivability Engineering, Technical Note CMU/SEI- 2003-TN033. Software Engineering Institute. Pittsburgh, Pennsylvania. 1-75.
4. Becker, S., Boskovic, M. & Dhama A. (2006, September). Trustworthy Software Systems: A Discussion of Basic Concepts and Terminology. Graduate School Trustsoft, Carl-von-Ossietzky University of Oldenburg, Germany.
5. Continuous Integration Strategies. Retrieved from http://developerblog.redhat.com/2013/11/08/ci-strategies-1of3/. Last visit 2016, March 3.
6. Becker, S., Boskovic, M. & Dhama A. (2006, September). Trustworthy Software Systems: A Discussion of Basic Concepts and Terminology. Graduate School Trustsoft, Carl-von-Ossietzky University of Oldenburg, Germany.
7. R. A. Khan, "From Threat to Security Indexing: A Causal Chain", Computer Fraud & Security, pp. 9-12, Vol 2009, Issue 5, May 2009.
8. Sahu K., Rajshree, "Software Security: A Risk Taxonomy", International Journal of Computer Science & Engineering Technology, pp- 36-41, 2015.
9. Continuous Integration Strategies. Retrieved from http://developerblog.redhat.com/2013/11/08/ci-strategies-1of3/. Last visit 2016, March 3.
10. Software for Dependable Systems: Sufficient Evidence. Retrieved from http://www.nap.edu/read/11923/chapter/4. Last visit 2016, March 2.
11. Kumar, R., Khan, S. A. & Khan, R. A. (2015). Revisiting Software Security: Durability Perspective, International Journal of Hybrid Information Technology (SERSC). 8(2), 311-22.
12. Kluwer, W. (2015, May 21). Starting your Software Security Assurance Program. ITARC. Stockholm, Sweden.
13. Kumar, R., Khan, S. A. & Khan, R. A. (2015, October). Durable Security in Software Development: Needs and Importance. CSI Communications, 34-36.
14. Rogers R. L. (2004). Principles of Survivability and Information Assurance. Carnegie Mellon University. Pittsburgh, Pennsylvania.

ABOUT THE AUTHORS

Mr. Rajeev Kumar is pursuing a Ph.D. in information technology from Babasaheb Bhimrao Ambedkar University (A Central University), Vidya Vihar, Raibareli Road, Lucknow. He has completed his master's degree in information technology from the same university. Kumar is a member of many national and international bodies, including ACM-CSTA, IBM-TechTarget, IAENG and BVICAM. His research interests are in the areas of software security, software durability and software risk. He is currently working in the area of software security durability.

Babasaheb Bhimrao Ambedkar Central University
Department of Information Technology
Lucknow-226025, UP, India
rs0414@gmail.com

Dr. Suhel Ahmad Khan has earned his doctoral degrees from Babasaheb Bhimrao Ambedkar University, (A Central University), Vidya Vihar, Raibareli Road, Lucknow. He is currently working as an assistant professor in the Department of Computer Application, Integral University, Lucknow, UP, India.

Dr. S. A. Khan is a young, energetic researcher and has completed a full-time major project funded by University Grants Commission, New Delhi, India. He has more than five years of teaching and research experience. He is currently working in the area of software security and security testing. He has also published and presented papers in refereed journals and conferences. He is a member of IACIT, UACEE and Internet Society.

Integral University
Department of Computer Application
Lucknow-226026, UP, India
ahmadsuhel28@gmail.com

Professor Raees Ahmad Khan has earned his doctoral degrees from Jamia Millia Islamia, New Delhi, India. He is currently working as a professor and Head in the Department of Information Technology, Babasaheb Bhimrao Ambedkar University, (A Central University), Vidya Vihar, Raibareli Road, Lucknow, India. Professor R. A. Khan has more than 13 years of teaching and research experience. His areas of interest are software security, software quality and software testing. He has published a number of national and international books, research papers, reviews and chapters on software security, software quality and software testing.

Babasaheb Bhimrao Ambedkar Central University
Department of Information Technology
Lucknow-226025, UP, India
khanraees@yahoo.com