Guest Editors' Introduction to the Special Section on Evaluation and Improvement of Software Dependability

Katerina Goseva-Popstojanova, *Senior Member*, *IEEE*, and Karama Kanoun

RELIANCE on the operation of software systems has increased tremendously over the years, spanning almost every aspect of our lives—from shopping and entertainment, to controlling medical devices that sustain human life, controlling airplanes, or managing the critical infrastructure. Advances in software engineering, combined with the fast increase of hardware performance and the widespread use of broadband network connectivity, have led to the development of an enormous number of useful applications. This explosion of functionality, however, often is achieved with larger and more complex software systems which pose new challenges to software engineers who need to develop highly dependable systems, while at the same time delivering the product on time and at a low cost.

Software dependability is the quality of the delivered service such that reliance can justifiably be placed on this service. The service delivered by a system is the system behavior as it is perceived by its users (computer or human). The notion of dependability integrates several attributes[1]: *reliability*: continuity of correct service; *safety*: absence of catastrophic consequences on the user(s) and the environment; *availability*: readiness for correct service; *confidentiality*: absence of unauthorized disclosure of information; *integrity*: absence of unauthorized alternation or deletion of information; and *maintainability*: ability to undergo repairs and modifications. Availability to authorized users only, combined with confidentiality and integrity are the three dependability attributes that comprise security. Obviously, safety and security are focused on specific classes of failures (i.e., catastrophic failures in the case of safety, and unauthorized access or handling of information in the case of security).

When a software fault is accidentally triggered by the provided inputs, users observe an accidental failure. On the other side, when a software fault is exploited intentionally by malicious attackers, it enables a successful security breach into computer systems and networks which can be treated as a malicious failure. Either way, achieving highly dependable software systems typically requires a combination of four sets of techniques: *fault prevention*, focused on preventing the introduction or occurrence of faults; *fault removal*, focused on reducing the number or severity of faults; *fault tolerance*, aimed at delivery of correct service in the presence of faults; and *fault forecasting*, focused on estimating the present number, the future incidence, and the likely consequences of faults.

The widespread use of software systems and their ever increasing size and complexity impose many challenges that the dependability research community, software developers, and quality assurance practitioners strive to meet. This special section addresses the current strong interest in developing improved means of specifying, designing, verifying, deploying, and maintaining complex software systems in contexts where high dependability is required and is crucial for system operation. The purpose of the special section is to compile a collection of papers that present novel theoretical and empirical research aimed at evaluating and improving different aspects of software dependability. The scope covers a wide range of methodologies, techniques, and tools applicable through different stages of the software life cycle. In response to our call for papers, we received 32 submissions, out of which 26 were considered to be relevant to the topics of this special section (i.e., evaluation and improvement of software dependability). Each of these 26 papers was reviewed by at least three expert referees. After two rounds of peer-review process, we selected four papers which present new findings on different aspects of software dependability.

Software testing plays a vital role in improving software dependability by detecting and fixing software faults before they manifest as field failures. The first paper in this special section addresses the problem of developing an approach for efficient and systematic testing of products in a software product line. In the paper "Incremental Test Generation for Software Product Lines," Engin Uzuncaova, Sarfraz Khurshid, and Don Batory specify each product as a composition of features, where each feature is specified as an Alloy formula. To ensure soundness of generation, the authors introduce an automatic technique for mapping a formula that specifies a feature into a transformation that defines incremental

1. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable and Secure Computing* vol. 1, no. 1, pp. 11-33, Jan.-Mar. 2004.

- *K. Goseva-Popstojanova is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, PO Box 6109, Morgantown, WV 26506-6109. E-mail: katerina.goseva@mail.wvu.edu.*
- *K. Kanoun is with LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse, France. E-mail: karama.kanoun@laas.fr.*

For information on obtaining reprints of this article, please send e-mail to: tse@computer.org.

refinement of test suites. Thus, the proposed approach of incremental test generation enables tests that are generated for one product to be used to generate tests for another product. Such reuse has great potential for optimized test generation, considering the large number of possible products in a software product line. The authors evaluate the proposed incremental approach to test generation using two subject product lines (i.e., binary trees and intentional names) and compare its performance with the conventional test generation based on the latest Alloy tool-set. Results show that for simple refinements, the performance of the incremental approach is comparable to the conventional approach, while for complex refinements, the incremental approach significantly outperforms the conventional approach. In addition, since the incremental approach solves the complete problem using subproblems that have significantly fewer variables and clauses, it holds potential for better scalability than the conventional approach.

The paper "Software Reliability and Testing Time Allocation: An Architecture-Based Approach" by Roberto Pietrantuono, Stefano Russo, and Kishor S. Trivedi presents an optimization model aimed at minimizing testing efforts to be devoted to software components so that they achieve a reliability level that can assure the required overall reliability of the software application. The proposed model takes into account the architecture of software application, including its interactions with the operating system. Further, the reliability of each component is assumed to grow with the testing time devoted to it, which results in using software reliability growth model for each component. The model also accounts for potential use of fault-tolerant mechanisms such as restart-component, retry-application, and failover to a standby and provides different levels of solutions based on the level of details of the information provided as input. The authors discuss two different ways to obtain such information: by design/code information and simulation before the testing or by dynamically profiling a real execution from system test cases of a previous version. The prediction accuracy of the model and its sensitivity to the variation of potential sources of errors are explored on a case study.

The next two papers deal with security related aspects of software dependability. Considering the effectiveness of security mechanisms at design stage is addressed in the paper "Verification and Trade-Off Analysis of Security Properties in UML System Models" by Geri Georg, Kyriakos Anastasakis, Behzad Bordbar, Siv Hilde Houmb, Indrakshi Ray, and Manachai Toahchoodee. The authors advocate the use of the Aspect-Oriented Risk-Driven Development (AORDD) methodology for developing secure systems and specifically focus on two analysis approaches that can be used at design time. First, they demonstrate how designers can formally verify that a security mechanism is effective in protecting against a given security breach. This is achieved by transforming a UML misuse model into Alloy and using the Alloy Analyzer to reason about security properties. The results of the analysis either give assurance that the security properties exist or show that they do not. The second analysis approach is focused on trade-off analysis that designers can use to compare alternative security mechanisms. This approach is based on Bayesian Belief Network topology, whose inputs consist of the evidence from the security analysis, risk information from other AORDD steps, and trade-off parameters such as project specific-goals, time-to-market, budget, laws and regulations, and business goals. The trade-off analysis computes a fitness score, showing how well the proposed security mechanism meets the project goals.

The last paper in this special section is focused on discovery of security vulnerabilities in already deployed open source or proprietary software. In "Vulnerability Discovery with Attack Injection," João Antunes, Nuno Neves, Miguel Correia, Paulo Verissimo, and Rui Neves present an attack injection methodology which adapts and extends classical fault injection techniques with a goal to automatically discover vulnerabilities in software systems. The proposed methodology, implemented in a tool called AJECT, consists of an attack generation phase followed by an attack injection campaign phase. It does not need the source code of the server to generate and inject the attacks. Instead, it uses the specification of the server's communication protocol and predefined test case generation algorithms to automatically generate a large number of attacks. During the attack injection campaign, the execution of the server in the target system and the responses returned to the clients are monitored. The collected information is later analyzed to determine if the server exhibited any abnormal behavior which would indicate that a particular attack was successful in triggering an existing vulnerability. The usefulness of the approach was assessed on 16 publicly available POP and IMAP servers. AJECT discovered vulnerabilities in five of these 16 fully patched up-to-date servers. Based on the results, the authors concluded that complex protocols are much more prone to vulnerabilities that simpler ones and that closed source, proprietary applications appear to have a higher predisposition to contain vulnerabilities.

ACKNOWLEDGMENTS

We would like to thank the authors of all of the submitted papers for their interest in the special section on evaluation and improvement of software dependability. The support of former *IEEE Transactions on Software Engineering* Editor in Chief Jeff Kramer and the IEEE Computer Society staff was invaluable in the preparation of this section. Last but not least, we would like to thank more than 85 experts on different aspects of software dependability who served as reviewers. Providing detailed reviews of submitted papers in a timely fashion contributed to the success of this special section.

Katerina Goseva-Popstojanova
Karama Kanoun
Guest Editors

Katerina Goseva-Popstojanova is a Robert C. Byrd Associate Professor in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, Morgantown. Her research interests are in reliability, availability, and performance assessment of software and computer systems, and computer security and survivability. She has published more than 70 journal and conference articles on these topics. Dr. Goseva-Popstojanova received the US National Science Foundation (NSF) CAREER award in 2005. She has served as a principal investigator on various NSF, NASA, WVU Research Corporation, NASA West Virginia Space Grant Consortium, CACC, and Motorola funded projects. She served as a program chair of the 18th International Symposium on Software Reliability Engineering. She has served and is currently serving on program and organizing committees of numerous international conferences and workshops. She is a senior member of the IEEE and a member of the ACM.

Karama Kanoun is Directeur de Recherche at LAAS-CNRS, heading the Dependable Computing and Fault Tolerance Research Group (http://www.laas.fr/~kanoun/). She was a visiting professor at the University of Illinois, Urbana Champaign, in 1998. Her research interests include modeling and evaluation of computer system dependability considering hardware as well as software, and dependability benchmarking. She has authored or coauthored more than 150 conference and journal papers, 5 books and 10 book chapters. She has codirected the production of a book on dependability benchmarking (Wiley and IEEE Computer Society, 2008). She is chairperson of the Special Interest Group on Dependability Benchmarking of the IFIP, and of the French SEE (Société de l'Electricite, de l'Electronique et des Technologies de l'Information et des Communications) Technical Committee on Trustworthy Computer Systems. She is vice-chairperson of the International Federation for Information Processing (IFIP) on Working Group 10.4 on Dependable Computing and Fault Tolerance.