Testability and Software Performance: A Systematic Mapping Study

Mohammad Mahdi
Hassan
Karlstad University
Karlstad
Sweden

mohammad.hassan@kau.se

Syed Muhammad Ali
Shah
Swedish Institute of Computer
Science, Stockholm, Sweden
shah@sics.se

Wasif Afzal
Mälardalen University
Västerås
Sweden
wasif.afzal@mdh.se

Sten F. Andler
University of Skövde
Skövde
Sweden
sten.f.andler@his.se

Birgitta Lindström
University of Skövde
Skövde
Sweden
birgitta.lindstrom@his.se

Martin Blom
Karlstad University
Karlstad
Sweden
martin.blom@kau.se

ABSTRACT

Software testability refers to the characteristics of an artifact that impact ease to fulfill test objectives. In most of the research on software testability, functional correctness of the software has been the focus while the evidence regarding testability and non-functional properties such as performance is sporadic. The objective of this study is to present the current state-of-the-art related to issues of importance, types and domains of software under test, types of research, contribution types and design evaluation methods concerning testability and software performance. We have conducted a systematic mapping study on the topic by following the recommended guidelines. We find that observability, controllability and testing effort are the main testability issues while timeliness and response time (i.e., time constraints) are the main performance issues in focus. The primary studies in the area use diverse types of software under test within different domains, with real-time systems as being a dominant domain. The researchers have proposed many different methods in the area, however these methods lack implementation in practice as suggested by our guides for research type, contribution type and design evaluation methods.

CCS Concepts

Software and its engineering ! Extra-functional properties;

Keywords

Systematic mapping study; software testability; Software performance

1. INTRODUCTION

While software testing dynamically verifies and validates that a program or a system behaves as expected, software testability refers to the degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met [31]. In other words, testability is a property of software that makes it easier to test and hence affects the effort needed for testing. The higher the testability is, the easier it is to perform testing activities such as designing, executing and analyzing tests.

Software testability has been investigated in several different dimensions. Freedman [9] defines a program as testable if it has no input-output inconsistencies and that it has the properties of observability (of outputs) and controllability (of inputs). A different interpretation of testability is given by Bache and Muller [1] where testability is determined by the coverage achieved by a test strategy such as branch coverage for control flow testing strategies. A probabilistic view on software testability is given by Voas and Miller [34] and Bertolino and Strigini [3], looking at the probability that the code will fail if it is faulty. In a majority, if not all, of these investigations on software testability, the functional correctness of the software has been or is assumed to be the focus. Little is known regarding what software testability issues impact non-functional properties.

In this paper, we investigate the relationship between software testability and another important non-functional property: software performance. Software performance is defined as the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage [31]. Software performance degradation is one of the primary problems reported by projects after field release [35]. Software performance

is also a critical concern in an embedded systems environment where resources are limited. We have performed an extensive systematic mapping study and have categorized the available evidence into testability and performance issues, types and domains of software under test, research type, contribution type and design evaluation methods used in relevant papers.

Our results show that conventional testability concerns of observability, controllability and testing effort are also major issues when software performance is being investigated. A bulk of software performance issues deal with the time factor (timeliness and response time). Different types of software under test are used such as general, control software and communication protocols, along with others. A variety of domains are represented with the domain of real-time systems being mostly represented. However, despite the presence of number of methods on testability and performance concerns, few papers evaluate them in practice.

The rest of this paper is organized as follows. Section 2 presents the method followed in conducting the systematic mapping study. Section 3 presents the different maps, thus answering the research questions. The results and threats to the validity of the study are discussed in Sections 4 and 5 respectively. Study conclusions are presented in Section 6.

2. METHOD

Kitchenham and Charters [20] define a systematic mapping study as a way to present 'a broad review of primary studies in a specific topic area that aims to identify what evidence is available on the topic'. After the need for a systematic mapping study has been identified, the most important step is the specification of research questions.

2.1 Research questions

In order to capture the existing views on testability and software performance, we have formulated the following research questions:

- RQ1: What are the different software testability and software performance issues addressed in existing studies?
- RQ2: What type of software under test is used and what domain is in focus in research on software testability and software performance?
- RQ3: What type of research, contribution type and design evaluation methods are represented in existing studies?

In terms of PICOC criteria for structuring research questions [20], our research question has no limitation with respect to 'comparison' and 'context' but has the following elements:

Population: software.

Intervention: testability and performance.

Outcomes: Issues of importance concerning testability and software performance, types and domains of software under test, types of research, contribution and design evaluation methods used.

Table 1: Count of papers before and after duplicate removal.

Source	Search count	After duplicate removal
Springer Link	9933	8551
IEEE Xplore	1161	748
ACM Digital Library	5683	3422
ISI Web of Science	617	578
Scopus	5103	4059
ScienceDirect	3651	1658
Wiley Online	5673	4343
Sub-total	31821	23359
Exact phrase search (IEEE Xplore, ACM digital library, Springer Link, ISI web of science and Scopus)	786	174
Total	32607	23533

2.2 Generating a search strategy

A search strategy is both an important and a necessary step in conducting an systematic mapping study. The search strategy was agreed upon after several rounds of trial searches using various combinations of search terms. Due to the broad scope of our research question, we finalized four search terms: software testability, software testable, software untestable and software non testable. These search terms were used separately in the following databases: Springer Link, IEEE Xplore, ACM digital library, ISI web of science, Scopus, ScienceDirect and Wiley Online Library.

This initial search was complemented with an exact-phrase search (in full-text/other fields) whereby the four search terms were used with double quotation marks. The exact phrase search was carried out in databases where this search option was available, IEEE Xplore, ACM digital library, Springer Link, ISI web of science and Scopus. We did not restrict the search results based on publication year as we wanted to be as inclusive as possible. Thus the default settings for the start year were used for each database. Table 1 shows the number of hits for each database. We got a total of 32607 papers after the initial and exact-phrase search. After duplicate removal based on title and abstract, we were left with a total of 23533 papers.

2.3 Study selection criteria

The purpose of study selection criteria is to identify primary studies that are relevant for answering the research questions. An important step in the study selection process is to list exclusion and inclusion criteria. We decided to exclude studies that:

do not relate to software engineering/computer science,

do not relate to software testability,

merely mention testability in a cosmetic/cursory manner, lacking any credible research on it,

have a focus on hardware/system testability (such as digital circuit testability analysis),

are book reviews,

are not written in English language,

are editorial papers written for special issues of different journals,

represent academic theses,

are books/book chapters,

are only discussing software testability without relating it to software performance.

We included all those studies that:

address software testability and its relation to software performance.

1. First a total of 2089 papers were discarded based on automatic removal by keywords. We removed papers with keywords that suggested them not to be relevant to software testability and falling in our exclusion criteria. Examples of such keywords include VLSI, microchips, CMOS, circuit design, cell array, voltage, transistor, ip op, microprocessor, nanometer, DRAM and SRAM.
2. The second step of the study selection involved reading the titles and abstracts of remaining 21444 papers and excluding papers not relevant to software testability. The papers were distributed among authors and for each paper we classified it as being either relevant, non-relevant or not clear, based on the stated exclusion criteria. Each paper was classified in this way by two authors. In case of disagreement among the two authors, the paper was marked as not clear. As a result of this step, we were left with 1422 not clear and 413 relevant papers.
3. The third step of study selection involved deciding on the not clear papers based on skimming the full-text of each paper to see if it relates to software testability. The skimming process for each paper was done in several steps: (1) reading the introduction and conclusion sections (2) searching for term testability in the full text and (3) reading sections if found relevant for decision-making. After the full-text skim, we were left with 807 relevant papers.
4. The fourth step of the study selection involved deciding on which of the software testability papers relate to software performance. We again skimmed the full-text of 807 papers, similar to the previous step, but now searching for software performance. After this full-text skim for software performance, we were left with 80 papers.
5. The fifth step of study selection was done to read full-text of the 80 papers. As a result of this step, we were left with 23 relevant papers.
6. The set of 23 relevant papers were complemented with additional 3 papers recommended by an expert on the subject. In the end, we had a total of 26 primary

studies for our systematic mapping study. The primary studies are listed in Table 2 with information regarding authors, year of publication and venue of publication.

2.4 Study quality assessment

The purpose of study quality assessment is to provide more detailed inclusion/exclusion criteria and to attach significance to individual studies during synthesis. We did not assess the quality of included studies using any pre-designed quality instrument. This was decided because of two reasons. First, our research question does not aim at finding the strength of inferences where study quality assessment is regarded as valuable. Second, we wanted to be as inclusive as possible when it comes to presenting the state-of-the-art.

2.5 Data extraction

The purpose of data extraction is to record information obtained from primary studies in a pre-designed data extraction form. The data extraction was done by four authors. Besides the general information about paper ID and title, the following specific information was gathered: (1) testability method/technique, (2) performance method/technique, (3) testability issue in focus, (4) performance issue in focus, (5) testability metric, (6) performance metric, (7) measured positive/negative impact of testability on performance, (8) type and domain of software under test, (9) type of research, (10) type of contribution and (11) design evaluation method used.

3. MAPPING OF STUDIES

In this section, the individual primary studies are mapped in different dimensions in order to answer our stated research questions (Section 2.1).

3.1 Issues of importance concerning software testability and software performance

We have divided the software testability issues discussed in our set of primary studies into following categories:

Observability (50%): the ability to observe output/internal states of a component or a software under test (Primary study IDs: P2, P5, P8, P9, P15, P17, P18, P19, P22, P23, P24, P25, P26).

Controllability (46.1%): the ability to control input and execution of a component/software under test as required for testing (Primary study IDs: P2, P5, P9, P15, P17, P22, P23, P24, P25, P26, P7, P20).

Automation (7.7%): the extent to which software testability aspects can be automated (e.g., using an automated testing framework and built-in tests) (Primary study IDs: P10, P11).

Testing effort (30.8%): the ability to reduce testing effort and to promote ease of testing (Primary study IDs: P18, P22, P7, P3, P4, P6, P12, P13).

Miscellaneous issues (15.4%): issues concerning testability and requirements traceability (Primary study

Table 2: List of primary studies.

ID	Ref.	Authors	Year	Title	Venue
P1	[2]	Beer, A., Heindl, M.	2007	Issues in testing dependable event-based systems at a systems integration company	Conference
P2	[23]	Kranitis, N., Xenoulis, G., Gizopoulos, D., Paschalis, A., Zorian, Y.	2003	Low-cost software-based self-testing of RISC processor cores	Conference
P3	[12]	Haller, K.	2013	Mobile testing	Journal
P4	[7]	Dias, O.P., Teixeira, I.M., Teixeira, J.P., Becker, L.B., Pereira, C.E.	2001	On identifying and evaluating object architectures for real-time applications	Journal
P5	[6]	Chanson, S.T., Loureiro, A.A.F., Vuong, S.T.	1993	On the design for testability of communication software	Conference
P6	[14]	Hierons, R.M., Kim, T.-H., Ural, H.	2004	On the testability of SDL specifications	Journal
P7	[28]	Salva, S., Fouchal, H.	2001	Some parameters for timed system testability	Conference
P8	[5]	Bozzano, M., Cimatti, A., Katoen, J.-P., Nguyen, V., Noll, T., Roveri, M.	2009	The COMPASS Approach: Correctness, modelling and performability of aerospace systems	Conference
P9	[16]	Izosimov, V., Guglielmo, G., Lora, M., Pravdelli, G., Fummi, F., Peng, Z., Fujita, M.	2012	Time-constraint-aware optimization of assertions in embedded software	Journal
P10	[26]	Merdes, M., Malaka, R., Suliman, D., Paech, B., Brenner, D., Atkinson, C.	2006	Ubiquitous RATs: How resource-aware run-time tests can improve ubiquitous software systems	Workshop
P11	[19]	King, T.M., Allen, A.A., Wu, Y., Clarke, P.J., Ramirez, A.E.	2011	A comparative case study on the engineering of self-testable autonomic software	Conference
P12	[24]	Limsoonthrakul, S., Dailey, M.N., Srisupundit, M., Tongphu, S., Pamichkun, M.	2009	A modular system architecture for autonomous robots based on blackboard and publish-subscribe mechanisms	Conference
P13	[11]	Hans-Gerhard G.	2001	A prediction system for evolutionary testability applied to dynamic execution time analysis	Journal
P14	[17]	Jevtic, M.S., Damjanovic, M.S.	1997	An approach to design for testability in hard real-time systems	Conference
P15	[22]	Kopetz, H., Zainlinger, R., Fohler, G., Kantz, H., Puschner, P., Schutz, W.	1991	An engineering approach to hard real-time system design	Conference
P16	[18]	Keshk, A., Ibrahim, A.	2007	Ensuring the quality testing of web using a new methodology	Conference
P17	[8]	Etkin, J., Zinky, J.A.	1989	Distributed debugging: Network analysis tools	Journal
P18	[33]	Vincent, J., King, G., Lay, P., Kinghorn, J.	2002	Principles of built-in-test for run-time-testability in component-based software systems	Journal
P19	[10]	Groce, A., Holzmann, G., Joshi, R.	2007	Randomized differential testing As a prelude to formal verification	Conference
P20	[25]	Lindstrom, B., O'utt, J., Andler, S.F.	2008	Testability of dynamic real-time systems: An empirical study of constrained execution environment implications	Conference
P21	[37]	Yingshi, X., Bin, L., Lian, R., Ping, X.	2006	A study on software architecture of testability experiment verification environment	Conference
P22	[29]	Schutz, W.	1991	On the testability of distributed real-time systems	Conference
P23	[32]	Thane, H., Hansson, H.	2001	Testing distributed real-time systems	Journal
P24	[4]	Birgisson, R., Mellin, J., Andler, S.F.	1999	Bounds on test effort for event-triggered real-time systems	Conference
P25	[21]	Kopetz, H.	1991	Event-triggered versus time-triggered real-time systems	Workshop
P26	[30]	Schutz, W.	1994	Fundamental issues in testing distributed real-time systems	Journal

ID: P1), testability in general (Primary study IDs: P14, P16) and testability verification (Primary study ID: P21).

It is clear that observability (50%) and controllability (46.1%) are the two most studied testability issues, followed by testing effort (30.8%). These percentages are also in line with what we expect of most studies on testability in general, i.e. not specifically related to software performance.

We have further divided the software performance issues discussed in our set of primary studies into following categories:

Response time (23.1%): the elapsed time between request generation and system response (Primary study IDs: P11, P3, P6, P12, P13, P1).

Timeliness (46.2%): the ability of a system to meet deadlines (Primary study IDs: P8, P9, P15, P18, P22, P23, P24, P25, P26, P7, P20, P14).

Memory usage (11.5%): the constraint on a function to be performed within specified memory limits (Primary study IDs: P19, P11, P1).

Miscellaneous issues (26.9%): the issues concerning overall system performance (Primary study IDs: P2, P5, P17, P10, P4, P16, P21).

The percentages of primary studies in each category of software performance issues clearly indicate that meeting time constraints (timeliness and response time) is the most important performance property under investigation, while resource consumption in terms of memory usage has received relatively less attention.

3.2 Type of software under test and domain

Figure 1 shows the frequency of type of software used in different primary studies. A variety of software under test have been used by authors, with "general" category used in 9 out of 26 primary studies (34.6%). This category refers to no particular type of software under test but rather spans to any software type within its domain. 3 out of 26 primary studies (11.54%) used "control software" as software type while same number of studies used "communication protocol". "Miscellaneous" software type refers to suites of test objects used; 2 studies used such software type. Primary studies P1 and P11 used two different types of software in their studies.

Similar to type of software under test, a variety of domains are represented in research on testability and software performance. "Real-time system" represents the domain most represented with 12 out of 26 primary studies focussing on it. "Aerospace domain" is represented by 2 primary studies while a number of other domains are represented with single studies. It is interesting to find a wide spread of domains represented, although not much research evidence is found in each one of them, with the exception of real-time systems. It is also evident that testability and software performance is a concern for more recent domains of "autonomic software", "autonomous vehicles", "ubiquitous software systems" and "mobile applications". Figure 2 shows the frequency of domain represented in research on testability and software performance.

3.3 Research type, type of contribution and design evaluation method

This section maps the primary studies into types of research, contribution and the use of design evaluation methods. This

Figure 1: Different types of software under test.

Figure 2: Number of primary studies in different domains.

Figure 3: Research type of primary studies.

is useful in determining how existing papers on the topic have approached the problem and what contribution do they constitute. Wieringa et al. [36] have presented a classification scheme for studies in requirements engineering which we find suitable to classify papers in this study. The classification scheme differentiates between the following research types:

Evaluation research: Investigation of a problem in practice or implementation of a technique in practice. The knowledge claims in such type of research are new knowledge of causal relationships among phenomenon or new knowledge of logical relationships among propositions.

Solution proposal: A solution to a problem is proposed, be it novel or a significant improvement of an existing technique. The proposal is accompanied by small example, a sound argument or by other means.

Validation research: Investigation of a solution proposal that has not yet been implemented in practice, e.g., experiments, simulations, prototypes, etc.

Philosophical paper: The paper presenting a new way of looking at things, e.g., a new conceptual framework.

Opinion paper: Author's opinion about what is wrong and good about something.

Experience paper: Author's personal experience of using a technique in practice that may not rely on discussion of research methods.

We categorized all papers in above types of research, shown in Figure 3. Two major categories in types of research are "validation research" (9 papers) and "solution proposal" (7 papers) respectively. This clearly indicates that most of the research results in the area lack implementation in practice, also indicated by only 6 papers in categories of "experience paper" and "evaluation research".

We further categorized the primary studies in terms of their research contribution type. We use the contribution facets given by Petersen et al. [27]: metric, tool, model, method

Figure 4: Contribution type of primary studies.

and process. The resulting map is shown in Figure 4. The top most contribution facet is "method" with 20 papers, followed by "model" (7 papers) and "process" (6 papers). Only 6 papers represent "tool" and "metric" categories. This map shows that while researchers have proposed methods/ techniques/ approaches, they have not been supported by tools and metrics. In light of Figure 3, this helps explain the lack of experience papers and evaluation research in the field.

We also categorized the papers with respect to their design evaluation methods, shown in Figure 5. This categorization is inspired by Henver et al. [13]. We classify papers into following design evaluation methods [13]:

Architecture analysis: Evaluating the fitness of the approach in technical architecture.

Informed argument: Building a convincing argument using relevant research.

Studying artifact in controlled environment for qualities (e.g., usability).

Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior.

Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility.

Simulation: Execute artifact with artificial data.

Case study: Study artifact in depth in business environment.

The top three design evaluation methods are informed argument (12 papers), architecture analysis (8 papers) and experiments (8 papers). 5 papers describe scenario-based evaluations. Few case studies (3 papers) and simulation studies (1 paper) have been conducted.

4. DISCUSSION

Our results have shown that testability and performance is an interesting combination where extensive evidence is lacking. We believe that it has to do with a general lack of research into performance issues (and for that matter into

Figure 5: Design evaluation methods of primary studies.

other non-functional properties) and also because testability is often ignored as an important concern during system design and development. Moreover, the terms software testability and software performance have multiple connotations. This creates a difficulty in designing search terms that capture every angle of topic under investigation. This is one of the reasons of starting broad in our search, with a focus on software testability, and then further narrowing the focus to software performance during study selection. The multiple connotations attached also creates a challenging task in synthesizing the available evidence since researchers take different research foci on the topic. We, therefore, believe that our aggregation of evidence under different categories of software testability and software performance is a useful contribution that can facilitate research with a defined focus. In Section 1, we briefly discussed the different existing interpretations on testability. In one of the earliest papers on program testability by Freedman [9], the author presented the idea on program testability in terms of observability of outputs and controllability of inputs. As our results indicate, these two properties of testability are also the most researched testability issues with respect to software performance. Moreover, a general notion of testability also relates to the ease with which testing can be done and to have reduced test effort. Our results also indicated evidence in this direction. Our results also showed an overall emphasis on time constraints (response time and timeliness) when investigating software performance. While this is not surprising since time is typically the attribute contributing mostly to performance [15], there are additional resource-usage scenarios impacting software performance such as throughput and capacity, memory usage and stability under workload. These aspects of performance have received little or no research with respect to testability. Our results also showed that a variety of domains are represented in research on testability and software performance, of special importance are newer domains such as autonomous vehicles, ubiquitous systems, autonomous software and mobile applications. This indicates that the scope of application of testability techniques for performance issues is widespread but lack implementation and evaluation in practice.

5. THREATS TO VALIDITY

There can be several threats to the validity of this study. Since a systematic mapping study claims to gather all available evidence regarding a topic of interest, the search process should be rigorous to ensure completeness. An obvious threat is that we might have missed including one or more relevant studies. Our search started broadly in order to not miss papers due to incorrectly formulated search strings. The decision to have a broad search was taken after a number of trial searches on two known databases and comparing the results to a set of known papers. We complemented the automated search with expert advice to ensure completeness of evidence. We also did not restrict our search with respect to time of publication. We, however, do not include grey or unpublished literature in this systematic mapping study. The study selection phase included multiple raters assessing every study for inclusion/exclusion. In case of a disagreement among two raters, a third person acted as an arbitrator. We did not undertake quality assessment of papers using a predefined quality instrument, and we argue in favor of this choice in Section 2.4. The data extraction form was designed with mutual discussion and by keeping the research question and possible extensions to this study in mind. The validity of data extraction was confirmed by using a subset of primary studies to extract data for the second time. The authors would also like to highlight that the mapping is limited to the information provided in the primary studies.

6. CONCLUSION

This paper is a systematic mapping study that has gathered the available research evidence on issues of importance, types and domains of software under test, types of research, types of contribution and design evaluation methods concerning research on testability and software performance. For software testability, the most researched issues are controllability, observability and testing effort while timeliness and response time are the most researched software performance issues. The software testability issues found are conventional testability issues researched elsewhere while for software performance, factors other than time such as memory usage and throughput are underrepresented. Testability and performance is a concern in many variety of software under test and domains, indicating a potentially much wider applicability. However, the research area lacks large-scale industrial studies to evaluate the proposed methods in practice.

Acknowledgment

The authors are thankful to Thomas Ostrand and Sigrid Eldh for involvement in early part of this study.

This work was funded by The Knowledge Foundation (KKS) through the project 20130085: Testing of Critical System Characteristics (TOCSYC).

7. REFERENCES

- [1] R. Bache and M. Mullerburg. Measures of testability as a basis for quality assurance. *Software Engineering Journal*, 5(2):86{92, 1990.

- [2] A. Beer and M. Heindl. Issues in testing dependable event-based systems at a systems integration company. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, 2007.
- [3] A. Bertolino and L. Strigini. On the use of testability measures for dependability assessment. *IEEE Transactions on Software Engineering*, 22(2):97{108, 1996.
- [4] R. Birgisson, J. Mellin, and S. F. Andler. Bounds on test effort for event-triggered real-time systems. In *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*, Washington, DC, USA, 1999. IEEE Computer Society.
- [5] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Nguyen, T. Noll, and M. Roveri. The COMPASS approach: Correctness, modelling and performance of aerospace systems. In B. Buth, G. Rabe, and T. Seyfarth, editors, *Computer Safety, Reliability, and Security*, volume 5775 of *Lecture Notes in Computer Science*, pages 173{186. Springer Berlin Heidelberg, 2009.
- [6] S. Chanson, A. Loureiro, and S. Vuong. On the design for testability of communication software. In *Proceedings of the 1993 International Test Conference (ITC'93)*, 1993.
- [7] O. Dias, I. Teixeira, J. Teixeira, L. Becker, and C. Pereira. On identifying and evaluating object architectures for real-time applications. *Control Engineering Practice*, 9(4):403{409, 2001.
- [8] J. Etkin and J. Zinky. Distributed debugging: Network analysis tools. *Microprocessing and Microprogramming*, 25(1^{a}AS5):307{312, 1989.
- [9] R. S. Freedman. Testability of software components. *IEEE Transactions on Software Engineering*, 17(6):553{564, 1991.
- [10] A. Groce, G. Holzmann, and R. Joshi. Randomized differential testing as a prelude to formal verification. In *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, Washington, DC, USA, 2007. IEEE Computer Society.
- [11] H.-G. Gro. A prediction system for evolutionary testability applied to dynamic execution time analysis. *Information and Software Technology*, 43(14):855{862, 2001.
- [12] K. Haller. Mobile testing. *SIGSOFT Software Engineering Notes*, 38(6):1{8, 2013.
- [13] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75{105, 2004.
- [14] R. M. Hierons, T.-H. Kim, and H. Ural. On the testability of SDL specifications. *Computer Networks*, 44(5):681{700, 2004.
- [15] IEEE Computer Society. Guide to the software engineering body of knowledge (SWEBOK) v3.0, 2014.
- [16] V. Izosimov, G. Guglielmo, M. Lora, G. Pravdelli, F. Fummi, Z. Peng, and M. Fujita. Time-constraint-aware optimization of assertions in embedded software. *Journal of Electronic Testing: Theory and Applications*, 28(4):469{486, 2012.

- [17] M. Jevtic and M. Damnjanovic. An approach to design for testability in hard real-time systems. In Proceedings of the 21st International Conference on Microelectronics (ICM'97), 1997.
- [18] A. Keshk and A. Ibrahim. Ensuring the quality testing of web using a new methodology. In Proceedings of the 2007 IEEE International Symposium on Signal Processing and Information Technology (SPIT'07), 2007.
- [19] T. M. King, A. A. Allen, Y. Wu, P. J. Clarke, and A. E. Ramirez. A comparative case study on the engineering of self-testable autonomic software. In Proceedings of the 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASE'11), 2011.
- [20] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.
- [21] H. Kopetz. Event-triggered versus time-triggered real-time systems. In A. Karshmer and J. Nehmer, editors, Operating Systems of the 90s and Beyond, volume 563 of Lecture Notes in Computer Science, pages 86{101. Springer Berlin Heidelberg, 1991.
- [22] H. Kopetz, R. Zainlinger, G. Fohler, H. Kantz, P. Puschner, and W. Schutz*. An engineering approach to hard real-time system design. In A. van Lamsweerde and A. Fugetta, editors, 3rd European Software Engineering Conference (ESEC'91), volume 550 of Lecture Notes in Computer Science, pages 166{188. Springer Berlin Heidelberg, 1991.
- [23] N. Kranitis, G. Xenoulis, D. Gizopoulos, A. Paschalis, and Y. Zorian. Low-cost software-based self-testing of RISC processor cores. In Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1 (DATE'03), Washington, DC, USA, 2003. IEEE Computer Society.
- [24] S. Limsoonthrakul, M. N. Dailey, M. Srisupundit, S. Tongphu, and M. Parnichkun. A modular system architecture for autonomous robots based on blackboard and publish-subscribe mechanisms. In Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO'08), 2009.
- [25] B. Lindstrom, J. O. utt, and S. F. Andler. Testability of dynamic real-time systems: An empirical study of constrained execution environment implications. In Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation (ICST'08), Washington, DC, USA, 2008. IEEE Computer Society.
- [26] M. Merdes, R. Malaka, D. Suliman, B. Paech, D. Brenner, and C. Atkinson. Ubiquitous RATs: How resource-aware run-time tests can improve ubiquitous software systems. In Proceedings of the 6th International Workshop on Software Engineering and Middleware (SEM'06), New York, NY, USA, 2006. ACM.
- [27] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08), Swinton, UK, UK, 2008. British Computer Society.
- [28] S. Salva and H. Fouchal. Some parameters for timed system testability. In ACS/IEEE International Conference on Computer Systems and Applications (CSA'11), 2001.
- [29] W. Schutz*. On the testability of distributed real-time systems. In Proceedings of the 10th Symposium on Reliable Distributed Systems (RDS'91), 1991.
- [30] W. Schutz*. Fundamental issues in testing distributed real-time systems. Real-Time Systems, 7(2):129{157, 1994.
- [31] Standards Coordinating Committee of the Computer Society of the IEEE. IEEE Standard Glossary of Software Engineering Terminology, 1990.
- [32] H. Thane and H. Hansson. Testing distributed real-time systems. Microprocessors and Microsystems, 24(9):463{478, 2001.
- [33] J. Vincent, G. King, P. Lay, and J. Kinghorn. Principles of built-in-test for run-time-testability in component-based software systems. Software Quality Journal, 10(2):115{133, 2002.
- [34] J. M. Voas and K. W. Miller. Software testability: The new veri cation. IEEE Software, 12(3):17{28, 1995.
- [35] E. J. Weyuker and F. I. Vokolos. Experience with performance testing of software systems: Issues, an approach, and case study. IEEE Transactions on Software Engineering, 26(12):1147{1156, 2000.
- [36] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requirements Engineering, 11(1):102{107, 2005.
- [37] X. Yingshi, L. Bin, R. Lian, and X. Ping. A study on software architecture of testability experiment verification environment. In Proceedings of the 1st International Conference on Maintenance Engineering (ICME'06). Science Press Beijing, 2006.