Matching context aware software testing design techniques to ISO/IEC/IEEE 29119

Matalonga, Santiago; Rodrigues, Felyppe; Travassos, Guilherme Horta

Published in:
Software Process Improvement and Capability Determination

DOI:
10.1007/978-3-319-19860-6_4

Published: 01/01/2015

Document Version
Peer reviewed version

Matching Context Aware Software Testing Design Techniques to ISO/IEC/IEEE 29119

Santiago Matalonga[1] , Felyppe Rodrigues, Guilherme H. Travassos[2]

[1] Universidad ORT Uruguay. Cuareim 1471. 11100 Montevideo, Uruguay
smatalonga@uni.ort.edu.uy.
2 Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
felyppers@cos.ufrj.br; ght@cos.ufrj.br

Abstract. A software system is context aware when it uses contextual information to help actors (users or other systems) to achieve their tasks. Testing this type of software can be a challenge since context and its variabilities cannot be controlled by the software tester. The ISO/IEC/IEEE 29119 intended to cover testing of any software system. It provides a common language and process for testing software systems, including a categorization of conventional testing techniques. This paper contains the initial results of our ongoing efforts to understand the testing of context aware software, Specifically, we evaluate whether the observed techniques for testing context aware software can be matched against the ISO/IEC/IEEE 29119 categories or if they represent a new breed of testing techniques. The results indicate that using conventional techniques variations to test context aware software systems does not produce evidence on their feasibility to test the context awareness features in such systems.

Keywords: Context awareness; Testing; ISO 29119:2013; Systematic Literature Review.

1 Introduction

The ISO/IEC/IEEE 29119 is a novel standard that covers software and systems engineering testing. It was published in 2013, and some parts have not been approved yet. The purpose of the series of software testing standards is to define an internationally- agreed set of standards for software testing that can be used by any organization when performing any form of software testing[1]. It means it should be suitable to adapt the testing techniques not only to different organizational scenarios, but also to keep pace with technological advances of software systems. Although relevant, its scope seems to be broad and ambitious, considering the reality of contemporary software.

Context aware software systems are a relatively new breed of software applications. As a result of the evolution of ubiquitous and pervasive computing, context aware applications take advantage of the myriad of sensors available in mobile devices and use that information to serve their actors (users or other systems). The verification and validation of this type of system is an area where software technologies have yet to be investigated [2]. Since the context of the application varies, testing all possible alternatives is not feasible. Context aware applications and their correspondent context states can explode the permutation of available test space for a software application. It is likely that coverage or other test completeness techniques might not be useful in the realm of context aware software system.

The ISO/IEC/IEEE 29119:2013, particularly part 4, defines software test design techniques. By taking into account the novelty of both domains, the ISO/IEC/IEEE 29119:2013 international standard and context aware software testing techniques identified during a *quasi* Systematic Literature Review [3], this paper describes the observed equivalence among Test Types and Test Design Techniques defined in the ISO standard and such testing techniques.

Considering the features of context aware applications, the results point out that most of the reviewed technical literature uses conventional testing techniques to test context aware software. In our opinion, this is an indication that either the testing of context aware software systems is no different from testing conventional software, or that there are still software technologies targeting at context aware software systems to be developed . We believe these results are important for the practitioners and software industry, because if they can only rely on traditional testing techniques to test context aware software systems, they are bound to keep taking tradeoff decisions between feasibility and coverage of their testing effort. It is unlikely that a conventional testing strategy can be comprehensive enough to execute all possible context variations throughout testing execution, increasing the risk of failures in such applications after they are delivered.

This article is organized as follows; Section II summarizes our research project in testing context aware software systems. Section **Error! Reference source not found.** presents a summary of the ISO/IEC/IEEE 29119:2013 family of standards. In Section **Error! Reference source not found.** the matching methodology and results are presented. Section **Error! Reference source not found.** discuss our findings and provide hypothesis for understanding the results. Finally we discuss the threats to validity in Section **Error! Reference source not found.** and summarize our conclusions in Section **Error! Reference source not found.**.

2 Acquiring Knowledge Regarding the Testing of Context-Aware Software Systems

Context Aware Testing for Ubiquitous Systems (CAcTUS) is an ongoing research project involving researchers from three universities (Universidade Federal do Rio do Janeiro, Universidade Federal do Ceará – both in Brazil – and University of Valenciennes and Hainaut-Cambresis in France). The objective is to understand test strategies for the quality assessment of actor-computer interaction in ubiquitous systems. The initial tasks for the CAcTUS' researchers are concerned with the identification of relevant knowledge in the area. Therefore, there are three teams undertaking secondary studies (*quasi* systematic literature reviews - qSLR [3]) to reveal evidence on testing techniques, test case design, test case documentation, and interoperability testing concerned with context aware software systems. Particularly, this paper presents some results of the qSLR whose objective was to identify techniques reported in the technical literature for testing context aware software systems For the sake of simplicity, only the main features of the qSLR protocol is going to be presented. A complete description can be obtained in [4]

The search string used was interactively organized by using the PICO approach [5]. Table 1 presents the components of the search string, which had been applied to the Web of Science, Scopus and IEEE Xplore databases. The use of these databases can be justified by their stability and coverage of technical literature they provide.

Table 1 QSLR'S SEARCH STRING

Population	Context Aware Software Systems
	"context aware" OR "event driven" OR "context driven" OR "context sensitivity" OR "context sensitive" OR "pervasive" OR "ubiquitous" OR "usability" OR "event based" OR "self adaptive" OR "self adapt"
Intervention	Software Testing
	"software test design" OR "software test suite" OR "software test" OR "software testing" OR "system test design" OR "system test suite" OR "system test" OR "system testing" OR "middleware test" OR "middleware testing" OR "property based software test" OR "property based software testing" OR "fault detection" OR "failure detection" OR "GUI test" OR "Graphical User Interfaces test".
Comparison	-
Outcome	Testing approaches
	"model" OR "metric" OR "guideline" OR "checklist" OR "template" OR "approach" OR "strategy" OR "method" OR "methodology" OR "tool" OR "technique" OR "heuristics"

From the 1680 different articles retrieved, 110 were selected after evaluating their title and abstract. These were then narrowed down to the 11 technical papers ([4], [5], [6], [7], [8], [9], [10], [11], [12], [13] and [14]) used in this work as input to evaluate the testing of context aware software in section IV.

3 Acquiring Knowledge Regarding the Testing of Context-Aware Software Systems

The ISO/IEC/IEEE 29119:2013 [1] represents a relatively new series of international standards whose purpose is to provide governance of processes of software testing. The series encompasses the following four parts:

- Part 1 purpose is to present the definitions of testing terms and some discussions on key concepts to the understanding of the ISO/IEC/IEEE 29119 series of software testing international standards [1].
- Part 2 purpose is to specify test processes that can be used to govern, manage and implement software testing for any organization, project or smaller testing activity. It comprises generic test process descriptions defining the software testing processes. Supporting informative diagrams describing the processes are also provided [15].
- Part 3 specifies software test documentation templates that can be used by any organization, project or smaller testing activity [15].
- Part 4 defines software Test Design Techniques that can be used during the test design and implementation processes defined in ISO/IEC/IEEE 29119-2 Test Processes [16]. This standard part is currently in ISO review process draft stage.
- Part 5 is in even early stages of the ISO review process. It will specifically target processes and techniques for keyword based testing.

A recurring topic through the different standards' parts is that the series of international standards were designed to be suitable for *any* type of organization, following *any* software development lifecycle. In this paper, we focus on Part 4: Test Design Techniques.

3.1 ISO/IEC/IEEE 29119-4-DIS. Test Design Techniques

The standard classifies the Test Design Techniques to achieve the testing process requirements (Part 2) in three broad categories:

1. Specification based techniques: the test basis (e.g. requirements, specifications, models or user needs) is used as the main source of information to design test cases.
2. Structure based techniques: the structure of the test item (e.g. source code or the structure of a model) is used as the primary source of information to design test cases.
3. Experience based techniques: Tester's knowledge and experience are used as primary sources of information to design test cases.

Table 2 presents ISO/IEC/IEEE 29119-4-DIS's Test Design Techniques grouped by these three categories.

Table 2 ISO/IEC/IEEE 29119 TEST DESIGN TECHNIQUES

Test Design Technique Category	**Test Design Technique**

Specification based testing	Equivalence partitioning;
	Classification tree method;
	Boundary value analysis;
	Syntax testing;
	Combinatorial test design techniques;
	Decision table testing;
	Cause-Effect graphing;
	State Transition Testing;
	Scenario Testing, and;
	Random Testing.

Structure based testing	Statement Testing;
	Branch Testing;
	Decision Testing;
	Branch Condition Testing;
	Modified Condition Testing, and;
	Data Flow Testing Test Design Techniques.

Experience based testing	Error Guessing

3.2 ISO/IEC/IEEE 29119-4-DIS and ISO 25010 Test Type

While ISO/IEC/IEEE 29119 defines the Test Design Techniques, it relies on a previously defined standard – the ISO/IEC 25010 Systems and Software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models – for the definition of Test Types.

As a quick introduction, the ISO/IEC 2510:2011 [17] is the current standard where quality attributes are defined. A quality attribute is a factor affecting the run-time behavior, system design and user experience. The ISO/IEC 2510:2011 defines Test Types aiming at the quality attribute the test is intended to evaluate. For the mapping purpose that will be presented in the following section, Table 3 presents the test types defined in this standard.

Table 3 ISO/IEC 25010 TEST TYPES

ISO/IEC 25010 Test Types	

Accessibility Testing;
Backup/Recovery Testing;
Compatibility Testing;
Conversion Testing;
Disaster Recovery Testing;
Functional Testing;
Installability Testing;
Interoperability Testing;
Localization Testing;
	Maintainability Testing;
	Performance-Related Testing;
	Portability Testing;
	Procedure Testing;
	Reliability Testing;
	Security Testing;
	Stability Testing, and;
	Usability Testing.

4. On Matching Testing Context Aware Software Systems Testing Techniques and ISO/IEC/IEEE 29119:2013

4.1 Matching Methodology

Mapping the Test Design Techniques, as it was initially our intention, was not possible since most of the identified authors did not make an explicit reference to the design techniques used in their work. Therefore, for our purposes we studied and named the techniques that, to the best of our knowledge, the authors seemed to have used (see 4).

In contrast, it was possible to extract names for the test types that the authors defined to be using (presented in Table 5). However, to map these types also represented a challenge, since the names they used to use do not have a literal correspondence to the types described in the standard. Therefore, we provided this matching by interpreting the intention of each test type in the standard to what the authors have reported.

For instance, Merdes et al. [9] define the test type as being "run time testing", while our interpretation is that run time is the means to achieve interoperability testing ("run time testing" is not a valid category in ISO/IEC/IEEE 29119). Meanwhile, She et al. [12] define that their tools provide functional testing, which we agree and maintained the authors classification in our matching.

4.2 Matching rationale

This section describes the 11 identified technical papers concerned with the testing of context aware software systems.

Alsos and Dhal [4] presents a case study for usability testing of context aware systems in healthcare. The observational cases were designed by physically designing new scenarios in their environment. They present three prototypes to be used by each participant in a sequential order. The definition of Scenario Testing according to the ISO/IEC/IEEE 29119-4 says that scenario testing uses a model of the sequences of interactions between the test item and other systems (in this context users are often considered to be other systems) for the purpose of testing usage flows involving the test item. Test conditions shall either be one sequence of interactions (i.e. one scenario) or all sequences of interactions (i.e. all scenarios)[16].

Merdes et al. [9] present a proposal for fidgeting with resource sensors in mobile devices at runtime. They present a XML-based-tool for helping with this intervention. This XML layer enables the user to configure run time scenarios where the Test suites for the application will be executed. The approach presented in the article and its practical examples are pre-established scenarios created with the proposed XML based-tool, similar to Alsos and Dhal[4].

Canfora et al. [7] present a case study where they recorded the users reactions to a mobile context aware application. Their aim was to build an automated workbench for mobile usability testing. The authors described their experimental design as a set of testing scenarios, in which each of them was created in order to attend a different user level of knowledge (normal user, smart user and businessman). This kind of approach is characterized as scenario testing.

Jiang et al.[6] and She et al. [12] presented an automated testing environment for testing mobile applications. Both environments allow the definition of context variables to be used in the test case design. Jiang et al.[6] separated the population on teams, all the teams were given the same tasks to be followed step-by-step (just as the previous presented scenario testing papers). However, in contrast to [6], She et al.[12] do not provide any indication of what is the technique the subject used to generate the test cases. The authors propose a framework to aid the testers and try to compare what they call “manual testing” with the presented approach. Nevertheless, none of these approaches were detailed, making the ISO/IEC/IEEE 29119-4 classification unfeasible.

In similar fashion, Amalfitano et al. [5] present an app running on the Android platform. This app juggles the input received by the device sensors and feeds it to the application under test. The authors define their Test Design Technique as “exploration-based technique”. The ISO/IEC/IEEE 29119-4 defines the Error Guessing Technique as the design of a checklist of defect types that may exist in the test item, allowing the tester to identify inputs to the test item that may cause failures, if those defects exist in the test item. Each defect type shall be a test condition. Considering Amalfitano et al. [5]'s features, we matched their approach to the error guessing Test Design Technique according to the ISO/IEC/IEEE 29119-4.

Ryan and Gonsalves[10] present an experiment to evaluate the usability of context aware applications in different device types. In this study, the authors adapt the same application to run on four different device types and then evaluate them for Usability related quality attributes (for instance, learnability and ease of use). Similarly to the previously presented scenario testing papers, the population group had a list of steps to follow during the experimentation.

Satoh[11] defines the aim of the study as interoperability test of context aware applications. The author’s strategy is to emulate several sensors thus creating a sandbox for the context aware application to run in. Even though the proposed emulator seems to be promising, the test descriptions were not detailed enough for making the matching against the ISO/IEC/IEEE 29119-4.

Tse et al. [13] coin the term “methamorphic testing” to describe their approach. The approach is based on specifying the context variables and sensors to which the application is aware, and then use mutation operators to juggle that initial specification in order to find defects. An initial database contains the input/output variables and the approach presented selects different combinations of inputs to check if the outputs remains the same in different input sequences of execution. ISO/IEC/IEEE 29119-4 defines Random Testing as a model of the input domain of the test item that defines the set of all possible input values. An input distribution for the generation of random input values shall be chosen. The domain of all possible inputs shall be the test condition for random testing.

Finally, Wang and Chan [14] propose a metric for evaluating how exposed a test suite is to context variables. Their approach hypothesizes that maximizing test suit development for that metric will improve coverage of the context of the application. Lu et al. [8] present a similar measurement concept, and provide a laboratory proof of concept of their approach that is used to help them identify challenges in testing context-aware software systems. Both of the authors try to observe the context-awareness testing problem as a structure-based problem, instead of specification-based (approach used by the previously cited papers). Even though testing all possible context variable combinations being unfeasible, the authors show different arrangements resulting in distinct coverage criteria for white-box testing. According to the ISO/IEC/IEEE 29119-4, Branch Testing is a control flow model that identifies branches in the control flow of the test item shall be derived. Each branch in the control flow model shall be a test condition. Even though the authors' proposals are different; both of them are executing branch testing.

4.3 Matching results

This section presents the results of identifying approaches to context aware software testing. Table 4 summarizes our classification of the sources against the standards Test Design Techniques, according to the rationale explained in the previous section.

Table 4: PAPERS CLASSIFICATION BY ISO/IEC/IEEE 29119 TEST DESIGN TECHNIQUE

Article	**ISO/IEC/IEEE 29119 part 4 Test Design Technique**
Alsos and Dhal[4]	Scenario Testing
Amalfitano et al. [5]	Error Guessing
Jiang et al.[6]	Scenario Testing
Canfora et al.[7]	Scenario Testing
Lu et al.[8]	Branch Testing
Merdes et al.[9]	Scenario Testing
Ryan and Gonsalves[10]	Scenario Testing
Satoh[11]	None
She et al.[12]	None
Tse et al.[13]	Random Testing
Wang and Chan[14]	Branch Testing

In contrast to the Test Design Technique classification, in Table 5 is possible to observe the comparison between the ISO/IEC 25010 (referenced standard for Test Types in ISO/IEC/IEEE 29119) and the authors own classification.

Table 5: COMPARISION OF AUTHORS' VS ISO/IEC'S TEST TYPE

Article	ISO/IEC 25010 Test Type	Authors defined Test Type
Alsos and Dhal[4]	Usability Testing	Usability Comparative Testing
Amalfitano et al. [5]	Procedure Testing	Exploratory Testing
Jiang et al.[6]	Compatibility Testing	None
Canfora et al.[7]	Usability Testing	User Experience Testing
Lu et al.[8]	Functional Testing	Functional Testing
Merdes et al.[9]	Interoperability Testing	Run-Time Testing
Ryan and Gonsalves[10]	Usability Testing Functional Testing	Usability Testing
Satoh[11]	Compatibility Testing Interoperability Testing	Interoperability Testing
She et al.[12]	Functional Testing	Functional Testing
Tse et al.[13]	Functional Testing	Metamorphic Testing
Wang and Chan[14]	Branch Testing	Coverage-based Testing

5 Discussions

The results in **Table** 5 show that five out of the 11 references use the same name for Test Type. Taking into consideration the novelty of the standard it could be an indication that the community did not have enough time to adopt the terms yet. In contrast, given that software testing is a long standing area of practice in Software Engineering, it is still striking that researchers are still using such diverse names to describe the same things.

Regarding the applicability of the proposed techniques for context-aware software testing, it seems clear that most authors have taken the approach of treating context awareness as another dimension which must be taken into account when designing test cases. This means, authors are using context variables as input, variables which should be defined in the test case and not influencing the testing scenario, according their perspectives.

With the exception of Satoh[11] and She et al. [12], all other authors are using Test Design Techniques that can be mapped to the Standard. Furthermore, the majority of the approaches presented are using Scenario Testing as its Test Design Technique, i.e. the approach guides the tester in which way and with what perspective it should execute the test. If we consider that the context variations may occur at any time and without control, whether the approach limits the way of testing, it is reasonable to believe that the approach will not support the test of context-awareness features.

However, since nine out of 11 studies can be classified regarding the international standard, it seems that either (a) testing context-aware software is no different from testing of conventional – not context aware – software, for this group of papers; or (b) the novelties of the issues associated with this specific type of software system (context awareness) have yet not been explored. We argue in favor of the second hypothesis, since it spite of having achieved nine classifications, this have not been straight forward. And in addition to this, there is no evaluation of the suitability of the test design techniques for the test of context aware software systems.

Another observation is that the authors in the reviewed technical literature have distinct perceptions of software testing. For instance, four out of 11 references do not include the need of an oracle or specification as an integral part of the construction of test cases. The international standards wording for the definition of test cases is rather ambiguous in this subject. The standard puts the focus on the dynamic execution of the unit under test with the aim of evaluating its quality, and gives the oracle a nice-to-have status.

6 Threats to Validity

We understand that there are two main threats to the validity of the discussions presented in this work. One is regarding the the draft status of part 4 of the ISO/IEC/IEEE 29119 international standard. In our experience with other ISO standards, there can be significant changes between the DIS and the final approved version of the standard. This means that it is likely that the Test Design Techniques classification might not be stable yet. In contrast, the Test Types are taken from the ISO/IEC 25010:2011 that is stable. On the other hand, it is also possible that we might have overlooked some concept during the matching process or that other researchers performing the same matching process can come up with different classifications for the same sources. This might have impact on the conclusions and results presented in this paper.

Another source of threat to validity is inherent to the research method applied for identifying the relevant technical literature. The qSLR research protocol [3] documents the research process and results, which for the sake of brevity are not discussed but mentioned here:

- Threats of missing literature and Selection Bias: both are always a threat in qSLR studies and are mitigated by a rigorous research protocol [3].

- Inaccuracy of data extraction and bias on synthesis of information: there is always the threats that we could have misunderstood the intention of the authors whose work we have study. The readers of this paper can evaluate the summary of this papers in the section 4.2.

7 Conclusion

In this paper we have presented the Standard for Testing Software Systems (ISO/IEC/IEEE 29119) and its matching with currently used testing techniques for context awareness software systems. The scope of this standard is broad and ambitious; it seeks to define processes and techniques for testing all types of software systems. Being context aware software systems a relatively new type of software, our aim with this paper was to evaluate whether this new international standard could cover the testing of context aware applications.

Therefore, this paper classified the identified technical literature on testing context aware software by using the ISO/IEC/IEEE 29119 Test Design Techniques and ISO/IEC 25010 Test Types. The relevant technical literature included in this work resulted from a qSLR undertaken in the context of CAcTUS project.

The testing of context aware applications brings on the challenges associated with an ever changing context in which the application is executed. Therefore, context can become an ever increasing source of possible input spaces for defining test cases.

The results presented in this paper show that most of the identified technical literature reports to be using traditional testing techniques to test context aware software and that they are using techniques which can be classified using the international standard. However, as far as we could investigate, we have not been able to identify a test design technique that specifically targets at context variables in its conception.

Nonetheless, the question still remains whether the reason this is so is that the challenges and issues associated with testing context aware software are still so new that they haven't been explored yet or that they are no different than the testing of regular (or non context aware) software. Nonetheless, we believe that further evidence is needed to evaluate this claim.

Acknowledgements

This work was supported by the "CAcTUS - Context-Awareness Testing for Ubiquitous Systems" project partially financed by CNPq – Universal 14/2013 Number 484380/2013-3. Prof. Matalonga is supported by Banco Santander Uruguay grant for young researchers.

References

1. Software and systems engineering Software testing Part 1:Concepts and definitions. ISO/IEC/IEEE 29119-1:2013. 1–64 (2013).

2. Malik, N., Mahmud, U., Javed, Y.: Future challenges in context-aware computing. Proc. IADIS Int. Conf. 306–310 (2007).

3. Rodrigues, F., Matalonga, S., Travassos, G.H.: Systematic literature review protocol: Investigating context aware software testing strategies. , Rio de Janeiro (2014).

4. Alsos, O.A., Dahl, Y.: Toward a best practice for laboratory-based usability evaluations of mobile ICT for hospitals. Proc. 5th Nord. Conf. Human-computer Interact. Build. Bridg. - Nord. '08. 3 (2008).

5. Amalfitano, D., Fasolino, A.R., Tramontana, P., Amatucci, N.: Considering Context Events in Event-Based Testing of Mobile Applications. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. pp. 126–133 (2013).

6. Jiang, B., Long, X., Gao, X.: MobileTest: A tool supporting automatic black box test for software on smart mobile devices. Proceedings - International Conference on Software Engineering (2007).

7. Canfora, G., Mercaldo, F., Visaggio, C.A., D'Angelo, M., Furno, A., Manganelli, C.: A case study of automating user experience-oriented performance testing on smartphones. Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013. pp. 66–69 (2013).

8. Lu, H., Chan, W.K., Tse, T.H.: Testing context-aware middleware-centric programs. Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering - SIGSOFT '06/FSE-14. p. 242. ACM Press, New York, New York, USA (2006).

9. Merdes, M., Malaka, R., Suliman, D., Paech, B., Brenner, D., Atkinson, C.: Ubiquitous RATs: How Resource-Aware Run-Time Tests Can Improve Ubiquitous Software System. 6th Int. Work. Softw. Eng. Middleware, SEM 2006. 55–62 (2006).

10. Ryan, C., Gonsalves, A.: The effect of context and application type on mobile usability: An empirical study. Conferences in Research and Practice in Information Technology Series. pp. 115–124 (2005).

11. Satoh, I.: Software testing for mobile and ubiquitous computing. Sixth Int. Symp. Auton. Decentralized Syst. 2003. ISADS 2003. (2003).

12. She, S., Sivapalan, S., Warren, I.: Hermes: A tool for testing mobile device applications. Proceedings of the Australian Software Engineering Conference, ASWEC. pp. 121–130 (2009).

13. Tse, T.H., Yau, S.S.: Testing context-sensitive middleware-based software applications. Proc. 28th Annu. Int. Comput. Softw. Appl. Conf. 2004. COMPSAC 2004. 458–466 (2004).

14. Wang, H., Chan, W.K.: Weaving Context Sensitivity into Test Suite Construction. 2009 IEEE/ACM International Conference on Automated Software Engineering. pp. 610–614. IEEE, Auckland (2009).

15. Software and systems engineering Software testing Part 2:Test processes. ISO/IEC/IEEE 29119-3:2013(E). 1–138 (2013).

16. IEEE Draft International Standard for Software and Systems Engineering--Software Testing--Part 4: Test Techniques. ISO/IEC/IEEE P29119-4-DISMay2013. 1–132 (2014).

17. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. ISO/IEC 25010:2011. 1–34 (2011).