Overview of Software Testing Standard ISO/IEC/IEEE 29119

Hesham Alaqail **Shakeel Ahmed**

King Faisal University, College of Computer Science and Information Technology,
Department of Computer Science, Saudi Arabia

Summary
Software testing is an essential part of software development cycle. It is considered an important activity where software is validated in compliance to requirements and specifications. Mostly, software testing is relevant to poor execution and documentation, causing additional burden on software companies or purchasers. Estimates show that 20% to 80% of total cost of software projects is spent on testing activities. Since no software can be perfect, the cost spent on testing activities is worthy especially in safety-critical systems. This paper provides an overview of ISO/IEC/IEEE 29119 software testing standard. The included parts of the standard are: concepts and definitions, test processes, test documentation, test techniques and keyword-driven test.

Keywords:
software testing; ISO standard; test process; ISO/IEC/IEEE 29119.

1. Introduction

Software testing is an essential part of software development cycle. Normally, to ensure the operation of any written piece of code, there must be a testing phase to discover whether the code reacts as intended [1]. The main aim of software testing is to detect defects. As per IEEE Standard 610.12-1990, software testing is defined as "The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items" [2]. In spite of the definition, testing cannot be a separate process from software development cycle. However, it is believed that software testing existed before the establishment of development life cycles. According to references, the early software testing activity was made at 1954. In reference to today's estimates, the cost spent on testing phase vary from 20% up to 80% particularly for critical-safety systems. The estimates are assumed to be for the full life cycle of the software systems. Historically, software testing was attached with high cost and poor coverage of testing standards professionally and academically.

New international software testing standard was formed by International standardization organization (ISO) and International Electrotechnical Commission (IEC) in the year 2013 [1]. ISO/IEC/IEEE 29119 is a standardization series novel which covers systems and software engineering testing. The first aim of such series is to create a set of standards that agreed internationally. The second aim is to have adaptable standards for different organizational scenarios that cope up with current technological advancement in software development [3]. The new international standard is envisioned to be more comprehensive by covering various level of testing processes. I addition, it is meant to cover documentation, design test techniques, testing vocabulary and concepts [1].

In this paper, an overview of ISO/IEC/IEEE 29119 pertaining software testing is given which focuses on some aspects such as concepts and definitions, testing processes, testing documentation, test design techniques and keyword-driven test. The rest of the paper is constructed as follow: fundamental software testing model, concepts and definitions, testing processes, testing documentation, test design techniques and keyword-driven test and conclusion.

2. Fundamental Software Testing Model

The new software testing model is based on a fundamental testing model where processes testing form the core of the model in addition to three more basic entities illustrated below in Figure 1. Documentation is generated as final result of performing processes test cases providing description of the output. Techniques entity is derived from requirements to be used for designing test cases. The used terminologies by other model's part are labeled as concepts and definitions.

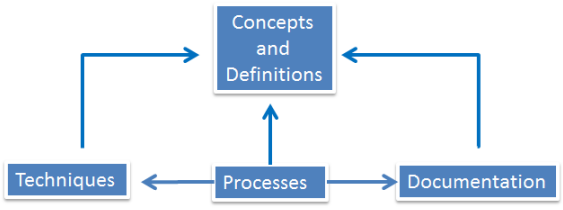

Fig. 1 New fundamental testing model.

The covered test process of the fundamental model executes at three different levels, those levels are illustrated at Figure 2 [1]. The organizational test process refers to the activities among several projects across the organization. Therefore, it contributes to the creation,

Manuscript received February 5, 2018
Manuscript revised February 20, 2018

development and implementation of organizational test policy. The test policy states management expectations of the organization as well as serving as direct approach for software testing in business terms.

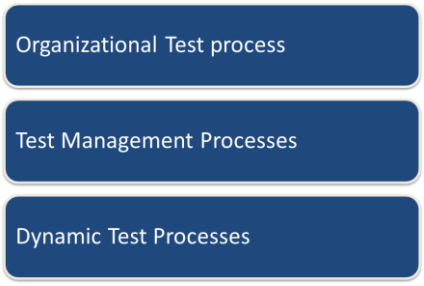

Fig. 2 Multilayer relationship diagram for test processes.

Requirements and constraints are dictated by the organizational test process to test management processes and dynamic test processes to engage with the organizational test process in all projects belonging to same organization. Formulation of new organizational test process is possible with different project natures that do not comply with the designed test processes. The comprehensive test stated by organizational test policy is divided into number of sub-processes of testing e.g. component, system, usability and performance testing. Test Management processes can be also applied to testing sub-processes. Different management plans can be used with testing sub-processes such as system, acceptance and performance test plans. Static and dynamic testing can be both included in testing the sub-processes [4].

3. Concepts and Definitions ISO/IEC/IEEE 29119-1

This is the first part of the ISO/IEC/IEEE 29119 standard. It focuses on key definitions and concepts of software testing for better understanding of the international standards. Software testing as concept is important for many reasons such as information regarding quality of the item being tested usually required by decision makers, the item under testing may not perform as expected and need to be verified and validated, evaluation should occur throughout the life cycle of the software being developed. Generally, it is acceptable that perfect software is impossible to achieve. Therefore, software continuous testing and evaluation is required. The main goal of testing is to provide information about the quality of items under testing. This information can be useful at following; removal of defects to improve the test item, improve decision made by management by considering the risk attached with items, enhance the organizational approaches and highlight the effective ones that reveal the defects [4].

4. Test Processes ISO/IEC/IEEE 29119-2

This part includes more details about test processes illustrated in Figure 2. It identifies the test processes that can be used for managing, governing and implementing software testing in organizations. It provides common descriptions of testing processes along with descriptive diagrams which applicable for all software testing models [5]. Figure 3 summarizes the processes and sub-processes involved in the multilayer relationship diagram. The organizational test is represented twice in Figure 3, once for the creation and maintenance of test policy and second for the organizational test strategy. Moreover, the test management process is commenced for the development and implementation of the test plan. The lower layer is dedicated for dynamic testing whenever needed by test plan. The dynamic testing for instance includes unit testing, system testing and performance testing, etc. [1].

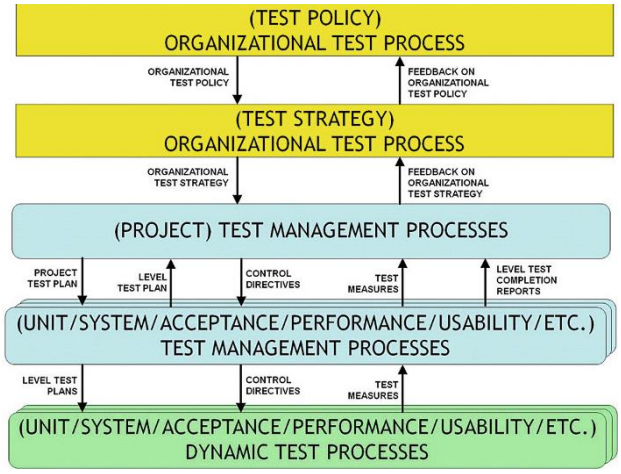

Fig. 3 Example use of multilayer model [1].

4.1 Test Monitoring and Control Process

Test monitoring and control process examines whether the progress of testing is in alignment with the organizational test plan and the test policy of the organization. If variance is detected, correction plan should be executed to remove the variance. This process can be assigned for the management of the project or the management of single testing phase or type e.g. System testing or performance testing. If the variance is detected at later stages, it would be applied as a part of dynamic testing. The person in charge of implementing test plan should implement the following activities and tasks in Table 1 [5].

Table 1: Activities and tasks example for test monitoring and control process.

Code	Activity Name	Task
TMC1	Set-Up	Create suitable measures and means to update risks with the new changes.
TMC2	Monitor	Collect and record measures, compare progress against test plan using measures collected.
TMC3	Control	Ensure implementing the activities required for test plan.
TMC4	Report	Test progress against test plan and communicate results to stakeholder within specific period of time, update and report new risks to stakeholders.

4.2 Test completion Process

Test completion process is a verification test to be performed when the test of activities is complete. It is used as verification test to carry out the testing done on system testing or performance testing to test the overall project. It is considered as satisfactory test leaving the testing environment in good situation. Communication to stakeholder should occur when overall testing is complete. Table 2 below summarizes the activities and tasks involved in this process [5].

Table 2: Activities and tasks example for test completion process.

Code	Activity Name	Task
TC1	Archive Test Assets	Properly identify and store useful test cases for later use.
TC2	Clean Up Test Environment	Restoration of test environment to pre-defined state upon completion of testing activities.
TC3	Identify Lessons Learned	Record lesson learned during the project.
TC4	Report Test Completion	Relevant information such as test plan, test results, test status report and test completion report should be collected and reported to stakeholders.

4.3 Dynamic Test Processes

This type of testing is used to carry out the dynamic testing in specific phase of testing e.g. unit, system, integration and acceptance testing. Four types of dynamic test process are available; test design and implementation, test environment set-up and maintenance, test execution and test incident reporting. These processes is normally considered as a part of test strategy implementation within test plan in the test phase e.g. System testing or type testing e.g. performance testing [5].

4.4 Test Design and Implementation Process

In this process, test cases and procedures are derived. Normally, documented in test specifications, but can be executed immediately. Possibly, stored test assets can be used during this process as regression test. This process can stop or reinitiate again in case of reporting new incident. Also, it requires tester to apply one or more testing techniques to derive test cases or test procedures aiming to attain test completion criteria. Moreover, iteration may occur among the activities. Table 3 summarizes the activities and tasks of this process [5].

4.5 Test Environment Set-Up & Maintenance Process

This process is used to create and maintain the environment where tests are performed. It may involve changes to test environment based on the result of the previous tests. The test environment may change depending on the configuration management processes. The tasks and activities within this process are; establishment of test environment and maintaining test environment [5].

4.6 Test Execution Process

This process is utilized to run the test procedures that are generated as result of test design and implementation process. Test execution process may require running several times since all available test procedures cannot run at single iteration. Furthermore, when an issue is fixed, the test execution process should be performed again. The activities involved within this process are; Test procedure execution, test results comparison and test execution recording [5].

Table 3: Activities and tasks for test design and implementation process.

Code	Activity Name	Task
TD1	Identify Feature Sets	Analyze test basis to understand the test item requirements, combine test features into test set and prioritize by risk value, document the feature set and communicate it to stakeholders.
TD2	Derive Test Conditions	Determine the test condition for each test case; prioritize the best condition based on risk, record test condition in test design specification.
TD3	Derive Test Coverage Items	Derive test coverage items through applying test design techniques; prioritize test coverage items based on risk, record test coverage items in test design specification. Record the traceability.
TD4	Derive Test Cases	Determine pre-condition and input values for one or more test cases and expected result prioritize by risk value, record test cases items in test design specification. Record the traceability, get approval of stakeholder.
TD5	Assemble Test Sets	Distribute test cases into one or more test sets based on constraints and execution, record test case in procedure specification, record the traceability.
TD6	Derive Test Procedures	Derive test procedures from ordered test set based on pre-condition, post-condition and dependencies, Identify excluded test data, prioritize test procedures based on risk, record test procedures in procedure specification, record the traceability, get approval of stakeholder.

4.7 Test Incident Reporting Process

This process is used for test incident reporting as a result of failure detection, items with unexpected or unusual behavior while executing the test or in case of retest. The activities involved are; analyzing test result and create or update incident results [5].

5. Test Documentation ISO/IEC/IEEE 29119-3

Test documentation part determines the software test forms and templates that can be used by organizations, specific project or single activity of testing. It contains definite documents that considered as an output of test processes. These documents can have multiple versions; such issue is related to configuration management [6]. The set of documentation is useful for test practitioners, below Table 4 list available documents in this part.

First two templates belong to organizational test process; the three proceedings belong to management test processes while the rest correspondent to dynamic test processes [1].

Table 4: Test documentation template list.

Seq.	Documentation name	Seq.	Documentation name
1	Test policy	**8**	Test procedure specification
2	Organizational test strategy	**9**	Test data requirement
3	Test Plan	**10**	Test environment requirement
4	Test status report	**11**	Test data readiness report
5	Test completion report	**12**	Test environment readiness report
6	Test design specification	**13**	Test execution log
7	Test case specification	**14**	Test incident report

6. Test Techniques ISO/IEC/IEEE 29119-4

This part specifies and identifies test techniques that can be used with test processes in part 2. The targeted audience of this part is testers, test managers and developers especially those who are in charge of software management and implementation. In this part, test design techniques are defined for specification-based testing, structure-based testing, and experience-based testing. In specification-based testing, the main source of information used to design test cases is test basis, for instance user needs, requirements, specification and models. In structure-based testing the source code or model structure is used as the source of information to develop test cases. In experience-based testing, the primary source of information is the experience and knowledge of testers. Moreover, all these types of tests are used to generate the expected end results. Those test design techniques are not essential but considered complementary. However, they are effective if applied in combination.

The terms specification-based testing, structure-based testing are also known as black-box testing and white-box testing (also named as clear-box testing), both black-box testing and white-box testing are related to the visibility of test items in software structure. In black-box testing, test item's visibility of the internal structure is not present while in white-box testing the internal structure of test items is visible. Grey-box testing is used as a term when knowledge of test items is utilized from specification and structure as well. Below Table 5 shows all techniques pertaining specification-based testing, structure-based testing and experience-based testing [7].

7. Keyword-Driven Testing ISO/IEC/IEEE 29119-5

Keyword-driven test is an approach of test specification which is generally used to aid test automation and creation of test automation framework. It can be used if no test automation approach is planned or exist. This type of testing can be applied to all level of testing such as component testing and system testing or even for test types like functional testing and reliability testing. The advantages of applying keyword-driven testing for system are; making system easier to use, understandable, maintainable, reusable, automation supportive, and money saver.

Table 5: Test techniques for specification and structure based testing.

Specification-based testing Techniques	Structure-based Testing Techniques	Experience-based Testing Techniques
Equivalence Partitioning	Branch / Decision Testing	Error Guessing
Boundary Value Analysis	Branch Condition Testing	
Cause-effect Graphing	Branch Condition Combination Testing	
Classification Tree Method	Data Flow Testing	
Combinatorial Test Techniques	Modified Condition Decision Coverage (MCDC) Testing	
Decision Table Testing	Statement Testing	
Random Testing		
Scenario Testing		
State Transition Testing		
Syntax Testing		

This part provides efficient definition and constraints for keyword-driven testing. It offers introduction and reference approaches to implement keyword-driven testing. Also, requirement definition for framework of keyword-driven testing is given. This enables testers to share their work items, test cases, test data, keywords or test

specifications. Moreover, this part provides interfaces definitions and data exchange format from different vendors to ensure different system integration. Furthermore, it includes definitions of different level of hierarchical keywords with explanation and advice on how to use them. Lastly, list of examples and technical low level keywords is provided e.g. “inputData” or “checkValue” to determine test case on any particular technical level. It can be also combined to from business-level keyword if needed [8].

8. Conclusion

Software testing is an essential part of software development cycle that cannot be neglected. Actually, it is an activity that should start upon the creation of software specifications until software deployment and further with changes and maintenance. The cost of adopting software testing approaches and practices is considerably high, but the final cost would be much higher in case the system encounters failure especially in enterprise and safety-critical systems. The awareness of testing standard is also beneficial to purchasers, businesses and governments to verify the quality of the market’s products on well-studied and measured basis.

Acknowledgment

I would like to thank my instructor Dr. Shakeel Ahmed for his assistance in understanding the testing firm in software quality during the academic course. Also, special thanks to my college at King Faisal University, College of Computer Science and Information Technology for facilitating all educational means to gain knowledge and excel.

References

[1] Reid S. 2012. “The New Software Testing Standard”. In: Dale C., Anderson T. (eds) Achieving Systems Safety. pp 237-255. Springer, London.

[2] Majchrzak T.A. 2012. “Software Testing”. In: Improving Software Testing. SpringerBriefs in Information Systems. pp 11-56. Springer, Berlin, Heidelberg.

[3] Matalonga S., Rodrigues F., Travassos G.H. 2015. “Matching Context Aware Software Testing Design Techniques to ISO/IEC/IEEE 29119”. In: Rout T., O’Connor R., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2015. Communications in Computer and Information Science, vol 526. pp 33-44. Springer, Cham.

[4] ISO/IEC/IEEE International Standard. 2013. “Software and systems engineering --Software testing --Part 1:Concepts and definitions," in ISO/IEC/IEEE 29119-1:2013(E) , pp.1-64. IEEE.

[5] ISO/IEC/IEEE International Standard. 2013. “Software and systems engineering --Software testing --Part 2:Test processes," in ISO/IEC/IEEE 29119-2:2013(E), pp.1-68. IEEE.

[6] ISO/IEC/IEEE International Standard. 2013. “Software and systems engineering -- Software testing --Part 3: Test documentation," in ISO/IEC/IEEE 29119-3:2013(E), pp.1-138. IEEE.

[7] ISO/IEC/IEEE International Standard. 2015. “Software and systems engineering--Software testing--Part 4: Test techniques," in ISO/IEC/IEEE 29119-4:2015, pp.1-149. IEEE.

[8] ISO/IEC/IEEE International Standard. 2016. “Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing," in ISO/IEC/IEEE 29119-5 First edition 2016-11-15 , pp.1-69. IEEE.

Hesham Alaqail earned his B.Sc. in Computer Science from King Faisal University, Saudi Arabia in 2009. He is pursuing his M.S. in same field at King Faisal University. Currently, Hesham is working as Programmer Analyst at Ministry of National Guard – Health Affairs, Saudi Arabia. His current research interests include Software Validation and Verification, Machine Learning and Classification Techniques.

Shakeel Ahmed received his B.Sc. (Computer Science in 1997) from Kakatiya University and M.C.A (Master of Computer Applications in 2000) from M.K. University, India. He earned his PhD degree in Computer Science from Indore University, India, in 2013. Currently he is working at King Faisal University, Saudi Arabia as Assistant Professor in the College of Computer Sciences and Information Technology. His current research interests include Mobile Ad hoc networks, Software Engineering, cloud computing he has published more than 15 papers in international journals and conferences in the field of software engineering and cloud computing.