

TEACHING SOFTWARE TESTING CONCEPTS USING LEARNING

OBJECTS

A Paper  
Submitted to the Graduate School  
of the  
North Dakota State University  
of Agriculture and Applied Science

By

Judi Lynn Simley

In Partial Fulfillment of the Requirements  
for the Degree of  
MASTER OF SCIENCE

Major Department:  
Computer Science

November 2016

Fargo, North Dakota

North Dakota State University  
Graduate School

---

**Title**

TEACHING SOFTWARE TESTING CONCEPTS USING LEARNING  
OBJECTIVES

---

**By**

Judi Lynn Simley

---

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**MASTER OF SCIENCE**

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia

---

Advisor

Dr. Kendall E. Nygard

---

Dr. Limin Zhang

---

Dr. Brian Slator

---

Approved:

December 12, 2016

---

Date

Dr. Brian Slator

---

Department Chair

## ABSTRACT

The ubiquitous nature of software, it is very important to prepare students for jobs in the software industry. There is a profound deficiency in testing skills in graduating students when beginning jobs in Information Technology Industry. This paper describes a means of teaching software testing concepts and tools in introductory computer courses using a Web-based Repository of Software Testing Tutorials: A Cyber-Learning Environment (WReSTT-CyLE). WReSTT-CyLE is a collaborative interactive eLearning environment for testing tools and concepts. The site was created to be student centered and to motivate testing concepts learning. This environment's design is customizable for Instructors' individual needs. Another aspect of WReSTT is the design of effective learning objects covering breadth of testing concepts. This paper's focuses on the creation of two learning objectives. The learning objectives focus on: System and Acceptance Testing levels. This paper covers testing importance, WReSTT design, and the detailed description of two learning objectives.

## ACKNOWLEDGEMENT

I would like to give thanks to all the people who supported me though out this entire process. That was started in 2010. I want to thank all of my Instructors and Professors that helped me along the way. I also want to thank all of the helpful people that support them and NDSU Departments. I give thanks to Dr. Slator for pushing me through this. I want to thank Dr. Nygard for helping me through both undergraduate courses and graduate courses. Thanks to Dr. Walia for guiding me through this paper and along with graduate courses. Dr. Zhang, thanks for helping with this process and teaching me in my undergraduate course.

I give thanks to my family members. To my mother for supporting me in many ways and giving me the attitude to not give up. To my father for supporting me and guiding me with the life skills to succeed. My sister for supporting my being! Most of all, I want to thank my husband, Scott. He has supported me though the complete process starting in 2004. He has helped with the raising of our 3 beautiful children, Alexis, Kali, and Race. I could not have done this without you! I love you all and THANKS!

## DEDICATION

I would like to dedicate this to my instructors, my parents, my family, and myself. I did it! I would also like to dedicate this to my husband and my children for all the absent hours mom was missing.

I dedicate this paper to all the Native American Indigenous Women; you can envision your achievements and succeed with whatever you do!

# TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENT.....	iv
DEDICATION.....	v
LIST OF FIGURES.....	vii
1. INTRODUCTION.....	1
2. BACKGROUND OF WReSTT.....	3
2.1. History of WReSTT.....	4
2.2 Using WReSTT-CyLE to improve software testing skills in IT students.....	5
3. LEARNING OBJECTIVES.....	7
3.1 Research path.....	7
3.2 Research data for Learning Objectives.....	7
3.3 Construction of Learning Objectives.....	9
3.3.1 System Testing LO.....	10
3.3.2 Acceptance Testing LO.....	25
4. FUTURE WORK.....	36
5. DISCUSSION AND CONCLUSION.....	37
REFERENCES.....	38
APPENDIX A. LISTING OF BOOKS REVIEWED.....	40
APPENDIX B. A LISTING OF WEBSITES VISITED.....	41
APPENDIX C. A LISTING OF CONFERENCE PAPERS REVIEWED.....	44
APPENDIX D. A LISTING OF ALL MATERIAL REVIEWED.....	46

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. The 1st page of System Testing LO .....	12
2. The 2nd page of System Testing LO.....	13
3. The 3rd page of System Testing LO .....	14
4. The 4th page of System Testing LO .....	15
5. The 5th page of System Testing LO .....	16
6. Continuation of the 5th page of System Testing LO.....	17
7. Continuation of the 5th page of System Testing LO.....	18
8. Continuation of the 5th page of System Testing LO.....	19
9. The 6th page of System Testing LO .....	20
10. The 7th page of System Testing LO .....	21
11. Continuation of the 7th page of System Testing LO.....	22
12. The 8th page of System Testing LO .....	23
13. Continuation of the 8th page of System Testing LO.....	24
14. The 1st page of Acceptance Testing LO .....	26
15. The 2nd page of Acceptance Testing LO.....	27
16. The 3rd page of Acceptance Testing LO .....	28
17. Continuation of the 3rd page of Acceptance Testing LO.....	29
18. The 4th page of Acceptance Testing LO.....	30
19. Continuation of 4th page of Acceptance Testing LO .....	31
20. The 5th page of Acceptance Testing LO.....	32
21. The 6th page of Acceptance Testing LO.....	33
22. The 7th page of Acceptance Testing LO.....	34
23. The 8th page of Acceptance Testing LO.....	35

# 1. INTRODUCTION

As IT Professionals, testing is one of the most sought out IT skill to have in any position in the Industry. Testing is considered the most important process to ensure quality software today [3]. Testing is a valued skill-set to have for any IT professional. As stated, “*If there is one new skill every QA professional needs today, it is this: the ability to write a test script*” [4]. This is not only important for QA professionals but all IT professionals. Employers are looking for some of these testing skills; analytical, technical, good verbal and written communication skills, productivity, ‘test to break’ attitude, detail orientation, willingness to learn and suggest process improvements, and a passion for software testing. [5] The advantage of having these skills does not only benefit the IT Industry but all other Global Industries; as well as Academia. All IT professionals have started as students; either learning through academia or on their own. There have been many studies; via employer/job positioning surveys and academic surveys that state there is a significant default in testing skills in students. [6] With this deficiency known; the Academic world has a great challenge. Although there are increasing amounts of testing tutorials and testing tools available on the web, there are not many that provide institutions with the student/instructor collaboration features. Web-based Repository of Software Testing Tutorials: A Cyber-Learning Environment (WReSTT-CyLE) is one of these opportunities [7]. WReSTT-CyLE is a collaborative learning environment of testing concept for students. The eLearning site is enhanced with gamification features, independency, and the ease of usability. Both Instructors and Students are motivated to participate within the website. The website was updated with gamified features. [8] This also enhanced the collaborative aspects. The website creators wanted to have complete independence. The user does not have to download any software to use it. The website’s design was also restructured. It was improved for easier user interactions. The heart of WReSTT-CyLE is the design of learning objectives (LO). These LO focus on specified testing concepts for example, Introduction to Software Testing – LO1. The LO



start at a basic level and proceed to dive into more depth information about the topic. [16][18] The LO design was based on the theory of using a linear learning path. [17][18] The paper introduces the importance of learning testing topics and focuses on teaching students these testing skills. It discusses the WReSTT-CyLE webpage designed specifically for this purpose. This paper goes over the creating of two learning objectives that were made for WReSTT-CyLE. It includes the future works to be completed. Then, the paper is summarized.

## 2. BACKGROUND OF WReSTT

The background of the paper focuses on how Web-based Repository of Software Testing Tools (WReSTT) came from an idea into an actual interactive site to teach student testing concepts and tools. The whole idea for the WReSTT site was to bring testing tools into the classroom with minimal interruption. It was a collaboration of Florida International University (FIU) and Florida Agricultural and Mechanical University (FAMU). It was supported by National Science Foundation and IBM Company. [9] The focus on testing tool tutorials was devised from the Industry. There were a lot of software bugs in the headlines. The industry needed software professionals with testing skills. WReSTT creators began to resolve this with the website. They decided to create a web repository of testing tools for students and instructors. This supported the development of quality software while in school by offering a program analyzer. It also provided forums for questions, a rating system for the quality of the testing tutorial, and more external links. [9] WReSTT was intended to support instructors with the latest and most informative tutorials to offer to students. The main focus at this time was for undergraduate students taking introductory to software development courses. These courses were chosen because most of the enrolled IT students would be first-timers to software programming. This was their first formal exposure to development and this was the best opportunity for having testing introduced. Also, since the instructors were focusing on development, this was a fantastic opportunity to involve a non-intrusive way to reveal testing. With all these ideas in mind, this is where WReSTT became palpable. The history gives a description of webpage and how it has evolved to present. The usage of WReSTT focuses on the how useful it is to student and Instructors.

## 2.1. History of WReSTT

WReSTT-CyLE is an extension of the WReSTT (V2) and WReSTT (V1). It is a TUES II project; a collaborative effort between Florida International University, Alabama AM University, Miami University - Ohio, and North Dakota State University. Web-based Repository of Software Testing Tools; known as WReSTT (V1) was introduced around 2009 and included 7 tools, software testing tutorials and links to other materials. [10] It was designed to support undergraduate computer science courses. Based on feedback from students and instructors; WReSTT (V2) introduced some enhanced features with gamification. It became a collaborative learning environment instead of a repository for learning materials. These features included access to student reports, the ability to create virtual teams for team projects, ability for instructors to load class rolls, and the ability for instructors to create course templates. This added gamification to the repository, the teams could compete and there were leader boards to track progress of the other teams. With the portal being down, the team decided to take the opportunity to enhance some of the features and created WReSTT-CyLE. They enhanced the learning objectives and added more topics, it just not based on tools. The focus is more on all testing concepts along with the testing tools. The team collaboration was enriched with more gamification. The website design was again enhanced for better viewing and for users to navigate easier. Also, the instructors' portal has more capabilities to monitor student performance. [6] WReSTT-CyLE is a collaborative learning environment that is non-intrusive to any classroom. This was a major point that has stayed with the whole progression of WReSTT/WReSTT-CyLE. The creators wanted it to be independent. There is no downloading of software—everything needed is right on the site. The site has grown into a whole learning experience. Instructors can pick which tutorials they want their students to learn along with learning objectives. They can monitor the students' progress more easily. The site is adaptable to the individual instructors' needs. The site was planned to be student-centric. The students don't need

to search for tutorials—all assigned tutorials are right there. They can view their progress and have automated test scoring. This means no wait time for quiz grades. They can collaborate with their team members and face-to-face meetings are only as necessary. With almost 50 universities currently using the site, the team also used this opportunity to enhance the websites security.

## **2.2 Using WReSTT-CyLE to improve software testing skills in IT students**

The best way to teach students is through motivation. To motivate students, there needs to be an environment that encourages involvement. One proven way to complete this is to have a Cyber-Enabled Learning environment with the addition of gamification. [8] WReSTT-CyLE was designed to incorporate both elements. Cyber-Enabled Learning or eLearning allows instructors access to alternative teaching avenues. They can introduce students to additional educational information not readily available before. A typical eLearning site involves having assignments and exercise submission. With today's technology, these sites are becoming not just text only sites but can be multi-media sites. [11] The site was created to be a supplement for a traditional classroom setting. The instructors use the site's learning objectives and tutorials in addition to their material. WReSTT-CyLE is a non-intrusive method to introduce students to testing concepts. Cyber-Enabled Learning is not enough to motivate most students alone. With the addition of gamification of some features of the eLearning environment, the site greatly improved student motivation. [11] What is gamification? Gamification is the process of adding game-like enhancements to ordinary tasks/activities to entice user participation. [12] So, why use gamification? There are many gamification studies proving that it offers a positive impact on users. [13] Gamification is used because it keeps the user involved and promotes user interactions. When adding game-like features the user do not 'feel' like they are learning; it's more like they are playing a game. This can be achieved by having a 'team' or by creating individual challenges. WReSTT-CyLE was updated with gamified features. These included; a leader-board, virtual team creation, virtual points (including

bonus points) for both teams and individuals, and discussion forums. The leader-board allows students to view the progress of other teams. [6] [8] Teams were formed by the instructor. They were given points for completing activities on the site in a timely manner. Individual points were also given out: these included; getting points for updating the student profile, interacting in the forums, accessing tutorials and submitting assignments/quizzes. These gamified features kept the students involved by challenging them to compete against the other teams in points. It also keeps them involved with interactions through the discussion forum and discussing the best way to get the most points. The demo website is located at <http://demo.wrestt.cis.fiu.edu/about-wrestt-com>. [7] The demo view included the Instructor and student views. The student view does not have as many options as the Instructors. The Instructor's view is customizable to the individual Instructor. Student can create their own student profile; this is the only customization they can perform. They have the same access (as Instructors) to all the tutorial and learning content. The heart of the site and main content are the learning objectives on the various testing concepts. Some of these have been added to the site and are under development. For this paper, the focus was creating two learning objectives; system testing and acceptance testing.

### 3. LEARNING OBJECTIVES

The learning objectives created were on system and acceptance testing concepts. This section describes the path that was taken to get to the research and the data collection for the learning objectives.

#### 3.1 Research path

Previous research on the WReSTT website about the gamification of the website and the guidance of my advisor, Dr. Gursimran Singh Walia, Ph.D., lead to the construction of this master paper. The research included reading a lot of borrowed books, intensive internet searching, gathering papers on the topic, and a lot of computing hours. The topic of this paper came up because Dr. Walia is the main contact at North Dakota State University (NDSU) for WReSTT-CyLE. The WReSTT-CyLE team wants to get more testing concepts onto the site. Dr. Walia provided a list of testing topics that were not on the site yet. All the other levels of testing were completed; the only high-level concepts left were system and acceptance testing. This was where the 2 learning objectives of this paper generated from. The complete listing of books can be found in Appendix A. The complete listing of the internet addresses can be found in Appendix B. All papers used are found in Appendix C.

#### 3.2 Research data for Learning Objectives

There is a lot of terms used for testing concepts. Most of them are used very loosely and some terms are used as double-meaning words. The hardest part of the data collection was the categorizing it. For example; ad-hoc testing was under testing methods and types of testing. Ad-hoc testing is a type of validation to test the software under test (SUT). It is an actual technique used. It should not be categorized with functional or non-functional. These are the categories of testing are what is being test; the type of requirement that is being tested. Ad-hoc testing is validating random pieces of SUT; therefore, can be functional or non-functional. Also, the concept mapping

between textbooks, webpages, and papers was very hard to follow. So, one of the greatest concerns was to categorize these terms correctly. The concept of manual or automated testing fell under the same principle as functional or black-box, in which a lot of the information grouped them together. There were also grouping of concepts and phases of testing, which didn't make sense either. All software testing is either functional or non-functional; these are the 2 categories of testing. Also, there are 4 different levels of testing: unit, component, integration, system, and acceptance. These were always followed in all information researched; the only difference was the naming. For example, system testing was sometimes named system-integration testing and acceptance testing was called user acceptance testing. All testing types, testing methods, and testing aspects can fall into any individual levels and any one of these categories. A compiled list with definition is found in Appendix D. The first grouping was types of testing were formed. These were based on the actual testing processes (how the testing was being performed) used for testing the Software Under Test (SUT). There are many different types of test: Active, Ad-hoc, Agile, Alpha, Automated, Beta, Big-bang, Big Bang Integration, Bottom-up, Bottom Up Integration, Concurrency, Dynamic, End-to-end, Exploratory, Hybrid Integration, Manual, Manual Scripted, Manual-Support, Model-Based, Negative, Pair, Passive, Parallel, Positive, Qualification, Regression, Requirements, Scenario, Static, Sandwich, System Integration, TDD, and Thread, Top-down, Top Down Integration, Upgrade, V-model, Waterfall (traditional). The next grouping was testing methods. This group was based on the information known or results of the tests that were being completed. There are different methods of testing: Assertion, All-pairs, Basis Path, Benchmark, Boundary Value, Black-box, Branch, Code-driven, Component, Condition Coverage, Context Driven, Decision Coverage, Destructive, Error-Handling, Equivalence Partitioning, Fault injection, Fuzz, Gorilla, Glass-box, Grey-box, Keyword-driven/ Table-driven, Loop, Modularity-driven, Mutation, Orthogonal array, Path, Statement, Smoke, Structural, Trial-error, White-box, and Workflow. A lot of these terms are

used inter-changeably in our industry; for example; glass-box and white-box testing. Both are testing methods with no real distinction. The final category was testing aspects. These are: Accessibility, Age, API, Backward Compatibility, Binary Portability, Breadth, Configuration, Compatibility, Compliance, Conformance, Conversion, Dependency, Domain, Endurance, Formal verification, GUI software, Globalization, Interface, Install/uninstall, Installation, Internationalization, Inter-Systems, Load, Localization, Operational, Penetration, Performance, Portability, Ramp, Recovery, Sanity, Scalability, Stability, Storage, Stress, Security, Upgrade, Usability, User Interface, Volume, and Vulnerability. This last grouping is based on validating the whole system as one. These are testing more of the non-functional requirements. For example, User interface testing is to validate how easily the user can perform tasks in the system. These are only some of the terms found in the research, there were many more. Now since the data was collected and categorized, the next step was to build the LO.

### **3.3 Construction of Learning Objectives**

It was very hard to cut out some of the information that was found. There was a lot of valuable information via textbooks, internet, and papers found. So, getting the correct amount of information was not an issue. It was cutting it down to be easily read on a webpage for the students using them and not to overload students. This editing was done to make the learning objective informative enough and to give a full understanding of the topic. Students that are accessing the site can have any level of understanding or exposure of testing concepts. This had to be kept in mind, so that a novice to someone with some knowledge can learn something from the LO. The last item was creating the learning objective to be aesthetically pleasing for the user. This was fulfilled by creating page breaks and adding visuals within the pages. Also, the design of the LO was based on a couple of concepts; linear paths and the layout of pages. [16] [17] The linear path concept is the first page is a basis and the consecutive pages become more in-depth in the topic. The layout of the page



itself was from a concept to keep the pages at a minimum for ease of use and visually appealing. [18] These items needed to be considered to create a great webpage for WReSTT-CyLE in order to support Instructors and teach students. The first LO was System Testing. This was the toughest one to create because it was the first page made. The first version was very word-heavy. There were 8 pages of text alone! It was submitted for review to Dr. Walia and Dr. Slator. Dr. Slator sent it back with comments and revisions. Dr. Walia commented that it needed to be shortened, there was too much text, and to add some visuals to make it appeal to the eye. He also sent some examples of other testing concepts. The LO was updated and sent back for final review. The compilation of Acceptance Testing LO was not much easier. Experience from the first LO helped with the creation. With the loosely used testing verbiage, it was a little trickier. Research was more in-depth with this LO. The path to a fully explained Acceptance Test was difficult to reach. Since there are many types of Acceptance testing, the main difference is the type of software created. There are 2 types of software created: custom-built and commercial-off-the-shelf (COTS). [14] To ensure the length of the LO stayed reasonable; only the custom-built software was considered. From the example and recommendation, visuals were added in the first version. It was sent to Dr. Walia for review and accepted. Both LOs were sent to peers for commenting.

### **3.3.1 System Testing LO**

The System Testing LO consists of a title, type of testing, testing methods, tools used, the basic concept, and quizzes. The title is System Testing – LO01. System Testing types are both manual and automated. The requirements that are tested are both functional and non-functional. This is the first time that both types of requirements are tested. Also, this will be the first time that the system is tested as an entire unit. [19] Some of the tools used are; LDRA, IBM Teleprocessing Network Simulator, and IBM Workload Simulator, test cases and testing scenarios. It contains 5 main concept pages and a page for references. These inform about the overview of System testing,

the differences between Integration and System testing, benefits, aspects of system testing, and steps taken in system testing. The LO is concluded with a Reference page. Also, there is a practice quiz and a real quiz to judge the knowledge of the students. The Overview Page includes the title at the top and some header information for the implementation of the page into WReSTT\_CyLE website. The main content is the background information of System Testing.

Figure 1 illustrates the 1st page of System Testing Learning Objective. This figure illustrates the Overview page of System Testing.

## System Testing – LO01

**Content:**

**Basic Concepts**

**Type: Manual and Automated**

**Method: Black-box, Functional, Non-functional**

**Tools: LDRA, IBM Teleprocessing Network Simulator, and IBM Workload Simulator, test cases and testing scenarios.**

### The Overview of System Testing

System testing is a level of testing that is performed after Integration Testing (see Integration Testing LO1) and before Acceptance (user) Testing (see Acceptance Testing LO1). It is the first time that the system is performing as a whole and tested. It is both a manual and automated process. The method of system testing is Black-Box (see Black-Box LO1). The tester does not know the “how” of the system, but only needs to know the expected output. For example, in a log-in situation, the user expects to be able to log-in with the correct credentials and not to log-in with the wrong credentials.

An analogy for system testing; producing a ball-point pen. During the process of manufacturing a ballpoint pen, the cap, the body, the tail, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed. [1]

Most of the test cases are derived directly from the requirements that are captured from the users before the software is written. They become use cases then eventually are used for test cases. This is how the test cases are formed. System Testing is the first testing level that will test both non-functional and functional requirements of the product. For the non-functional side of system testing, the test categories would be performance/load, scalability, reliability, stress, interoperability, localization, security, portability, installation, and usability. The functional side of system testing has a focus on the real-life user perspective of the product and the business needs of the organization. Functional test cases focus on the behavior of the product; they will include specific execution steps within the test cases. [2]

This level of testing is not used for debugging. Debugging is mainly to find and fix errors or bugs in the software. This testing level is mainly to check to see if the test cases pass or not. System testing uses a check list format. Test cases are applied and marked off if they are passing within the parameters of the specified requirement. Although bugs are tracked in System testing, they are not fixed. The bug is reported and then given back to another testing team to correct. System testing is usually performed by specified testing teams. Some companies have release management teams that would perform this or quality management teams

<new page>

Figure 1. The 1st page of System Testing LO

The second page of the System Testing Learning Objective gives a listing of the differences between Integration and System Testing. Integration is the second level of testing and System is the third level. The two levels are very close in distinction, but are very different in what the testing focus is.

**The difference in Integration and System Testing Levels**

The difference between the System and Integration testing levels are: Integration Testing is performed on integrating modules of the system. The modules are implemented and then tested one by one. Then the system integration tests are completed. Only after this the System testing is started. System testing is the validation and verification of the whole system. All modules are implemented as a whole in the testing environment that is like the real-world environment.

Here are some of the differences in table format for better viewing:

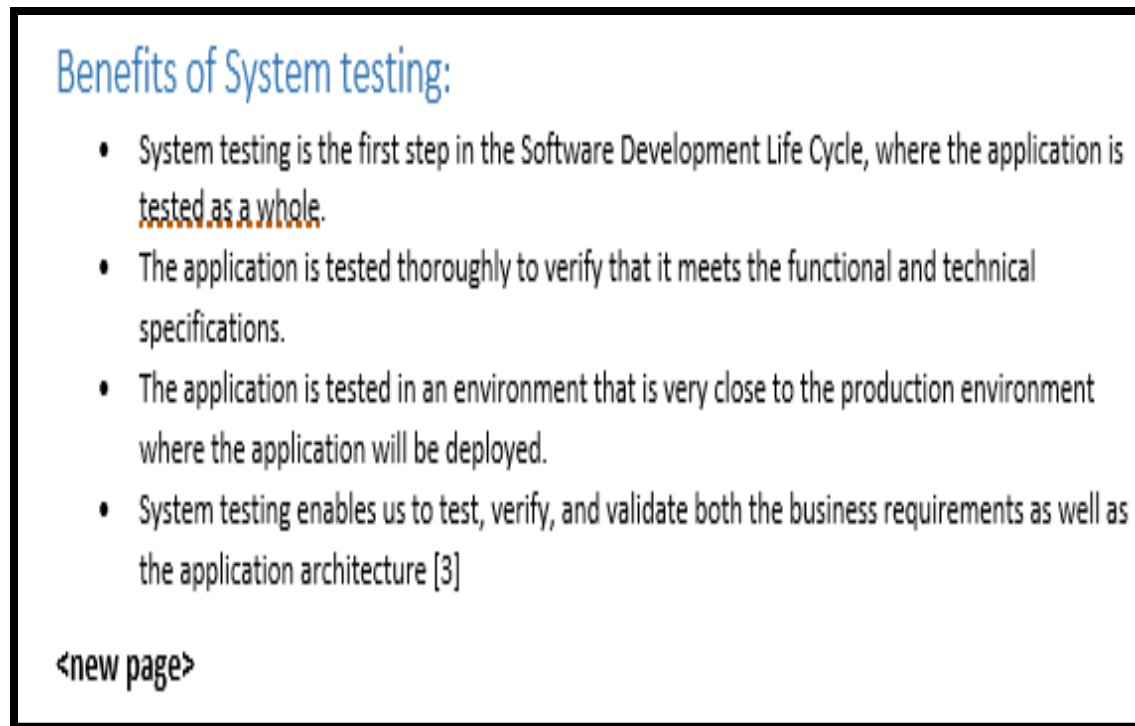
<b>System Testing</b>	<b>Integration Testing</b>
1. Testing the completed product to check if it meets the specification requirements.	1. Testing the collection and interface modules to check whether they give the expected result.
2. Both functional and non-functional testing are covered like; sanity, usability, performance, stress, and load.	2. Only Functional testing is performed to check whether the two modules when combined give correct outcome.
3. It is a high-level testing performed after integration testing.	3. It is a low-level testing performed after unit testing.
4. It is a black box testing technique so no knowledge of internal structure or code is required.	4. It is both black box and white box testing approach. So, it requires the knowledge of the two modules and the interface.
5. It is performed by test engineers only.	5. Integration testing is performed by developers as well test engineers.
6. Here the testing is performed on the system <u>as a whole including</u> all the external interfaces, so any defect found in it is regarded as defect of whole system.	6. Here the testing is performed on interface between individual module; <u>thus</u> any defect found is only for individual modules and not the entire system.
7. In System Testing the test cases are developed to simulate real-life scenarios.	7. Here the test cases are developed to simulate the interaction between the two modules.

*Figure 1.* Table of the differences between Integration and System testing.  
<http://www.softwaretestingclass.com/difference-between-system-testing-vs-integration-testing/>

<new page>

*Figure 2.* The 2nd page of System Testing LO

The third page of System Testing includes the benefits of system testing. The list was shortened because the other entries were related to the same items listed.



**Benefits of System testing:**

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture [3]

<new page>

*Figure 3.* The 3rd page of System Testing LO

The fourth page of System testing is the Aspects in System Testing. The page gives some of the examples used for system testing and then guides on how to choose from those examples.

**The aspects of system testing:** Functional, Hardware/Software, Load, Migration, Recovery, Regression, and Usability are some of the main aspects. There are more than 100 testing aspects that can be used with System testing. [4]

To determine which testing aspect to choose; there are some questions to ask.

First, who does the tester work for? This is important because larger software companies will have different objectives than medium to smaller companies. The company testing philosophies will vary also.

Second, how much time is available? This will help figure out how many tests you can complete within the schedule. This may be changed along with the schedule throughout the Software Development Life Cycle (SDLC).

Third, how many resources are available to test? How many 'hands on deck' will tie in with the time available. There are automated tests and manual tests, and this will help find which ones are suited for the testing need.

Fourth, how much do the testers know? The knowledge of the testers will help figure out what tests the testers can perform. Also, they will need to learn how to use the testing software if not already known. There are different aspects of testing and different testing processes. The timings of when to test a module and what to look for in the outputs are a huge factor in choosing the right test.

Fifth, how much money is there for testing? This is always the bottom line. This will determine which testing processes will be used. The testers will need to either use what they have for testing tools or choose to purchase them. This is a major factor for how many testers are on the project. [5]

<new page>

Figure 4. The 4th page of System Testing LO

The fifth page of the learning objective is the steps taken to complete System Testing. The page turned out to be larger than the rest because of the illustrations and the information gathered to give a full understanding of the concept. There are 7 steps all together. The Illustration is of the first step to take when starting System testing,

**In Software System Testing following steps needs to be executed:**

First, the System Test Plan is prepared. The points covered will vary from company to company. The project plan, test strategy and main test plans will also influence these. These are some main points in a System Test Plan:

- Goals & Objective
- Scope
- Critical areas Area (to focus)
- Test Deliverable
- Testing Strategy for System testing
- Testing Schedule
- Entry and exit criteria
- Suspension & resumption criteria for system testing
- Test Environment
- Roles and Responsibilities
- Glossary Example of a generic test plan:

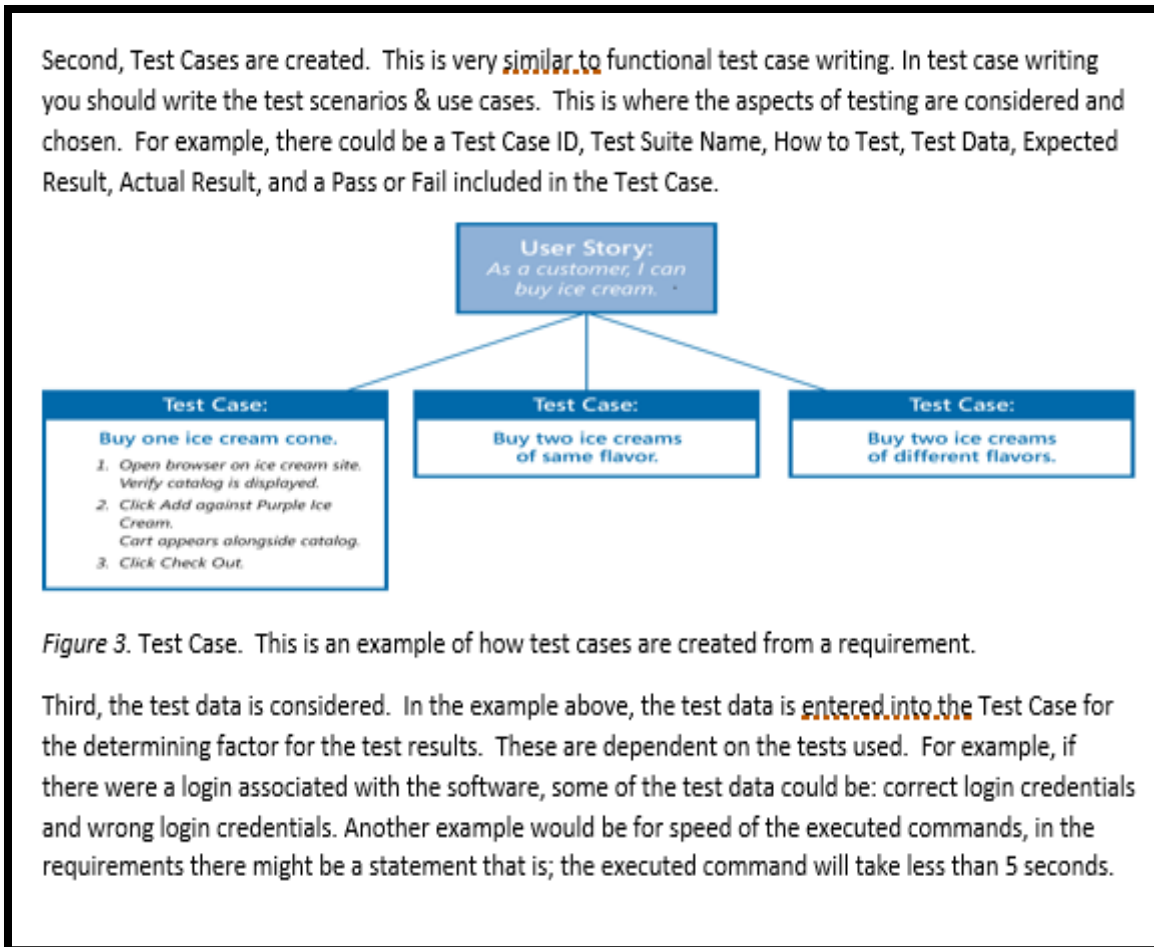
**Simple Test Plan Template**

Project Name: \_\_\_\_\_ Indicate name of project, application, UI, or fix  
 Testing Services Requested By: \_\_\_\_\_ Indicate individual or department requesting testing to be performed  
 Date Requested (Today's Date): \_\_\_\_\_ Indicate date of the above request  
 Technical Director: \_\_\_\_\_ Indicate appropriate director  
 Project Manager: \_\_\_\_\_ Indicate project and/or program manager (if any)  
 Test Lead: \_\_\_\_\_ Indicate the Test Lead (your name)

<b>Project Information</b>		
<b>Objective of Project:</b> State the purpose and objective of the project. What functionality is being created, added, and/or modified to what application/system and why?		
<b>Documentation Available (if any):</b> List any project-related documentation (and network directory location), such as business requirements, functional specification, and/or technical design.		
<b>Summary of Testing Required:</b> Summarize what needs to be tested in terms of functionality, components, business processes, and/or data flows.		
<b>Test Environment:</b> List the environment where testing is to be performed.		
<b>Test Data Required:</b> Specify the test data necessary to perform testing		
<b>Tools Required:</b> Specify tools that will be used to complete testing, such as Test Director or WinRunner.		
<b>Testing Key Contacts and/or Subject Matter Experts:</b> Identify and list key contacts and/or subject matter experts that are essential to complete testing		
<b>Special Testing Requirements:</b> Indicate any unique testing requirements and/or instructions.		
<b>Project Timeline</b>	<b>Start Date</b>	<b>End Date</b>
Development	Enter development start date.	Enter development end date.
Product Testing: Staging Environment	Enter start date.	Enter end date.
Performance Testing: PLT Environment	Enter start date.	Enter end date.
Product Testing: Production Environment	Enter start date.	Enter end date.
Estimated Launch Date:	Enter estimated launch date.	

Figure 5. The 5th page of System Testing LO

The sixth illustration shows the second and third steps to complete the testing. It illustrates how the test cases are derived from the user story.



*Figure 6. Continuation of the 5th page of System Testing LO*



The seventh figure illustrates the fourth step of the process. It gives a test case example along with the output of the test.

Fourth, the Automated test case execution is completed. A lot of the automated tests are already created from the Unit and Integration testing levels. These tests are combined into larger testing suites and then executed.

Output of an automated test case example:

This example is for a 3D car software game. In this automation software, the test data is entered. For this example, the test case is for loading the game. The exact steps are provided into the software and the expected results are given. The automated test cases are performed by the specialized testing team before User Acceptance Testing is started. These will simulate the real-life scenarios of the users.

ID: 1		Test Case: Load a game	
Domain	Game X Tests		
Type	Usage		
Priority	Med		
Title	Load a game		
Assumptions	<ul style="list-style-type: none"> <li>A game has previously been saved.</li> <li>The application is running.</li> </ul>		
Steps	<ol style="list-style-type: none"> <li>Select 'Load Game' from the main application menu.</li> <li>Click on the desired saved game from the list.</li> <li>Click on the 'Load' button.</li> </ol>		
Expected Results	<ul style="list-style-type: none"> <li>The load game screen should show all the saved games in the save game folder.</li> <li>The game should be loaded with all elements in the exact state they were left at the time of saving.</li> <li>The application should be in player turn state with the world view screen visible.</li> </ul>		
Variations	None		
Update			

Figure 4. Example of Automated Test Case. This example is from a Project management tool called [ProjectLight](#).

Then, when the test case is executed this was the output:

Testing	ID	Test Case	Type	Priority	Status	Build	Date	Issue	
Test core game features	1	Load a game	Usage	Med	Pass	100	2008/12/11		+
					Pass	100	2008/12/19		Log
					Pass	100	2008/12/19		
	2	Auto Save and Restore	Usage	Med	Pass	100	2008/12/11		+
					Pass	100	2008/12/19		Log
					Pass	105	2008/12/19		
	3	New game creation	Usage	Med	Pass	100	2008/12/11		+
					Pass	100	2008/12/11		Log
Test network play	4	Game 3D rendering - land and sea	Usage	Med	Fail	100	2008/12/11	26	+
					Pass	105	2008/12/19		Log
	5	Game 3D rendering - cities	Usage	Med	Pass	100	2008/12/11		+
					Pass	100	2008/12/11		Log
	6	Game 3D mouse navigation	Usage	High	Fail	100	2008/12/11	27	+
					Pass	100	2008/12/11		Log
	7	Move unit command	Usage	Med	Untested				+
	8	Unit work land command	Usage	Med	Untested				+
	9	Network game setup	Usage	Med	Untested				+
	10	Network game play	Usage	Med	Untested				+
		<b>10 Test Cases</b>							

Figure 7. Continuation of the 5th page of System Testing LO

The figure illustrates the 5<sup>th</sup>, 6<sup>th</sup>, and 7<sup>th</sup> steps in the testing process. It has a diagram of the flow of bug reporting that is used.

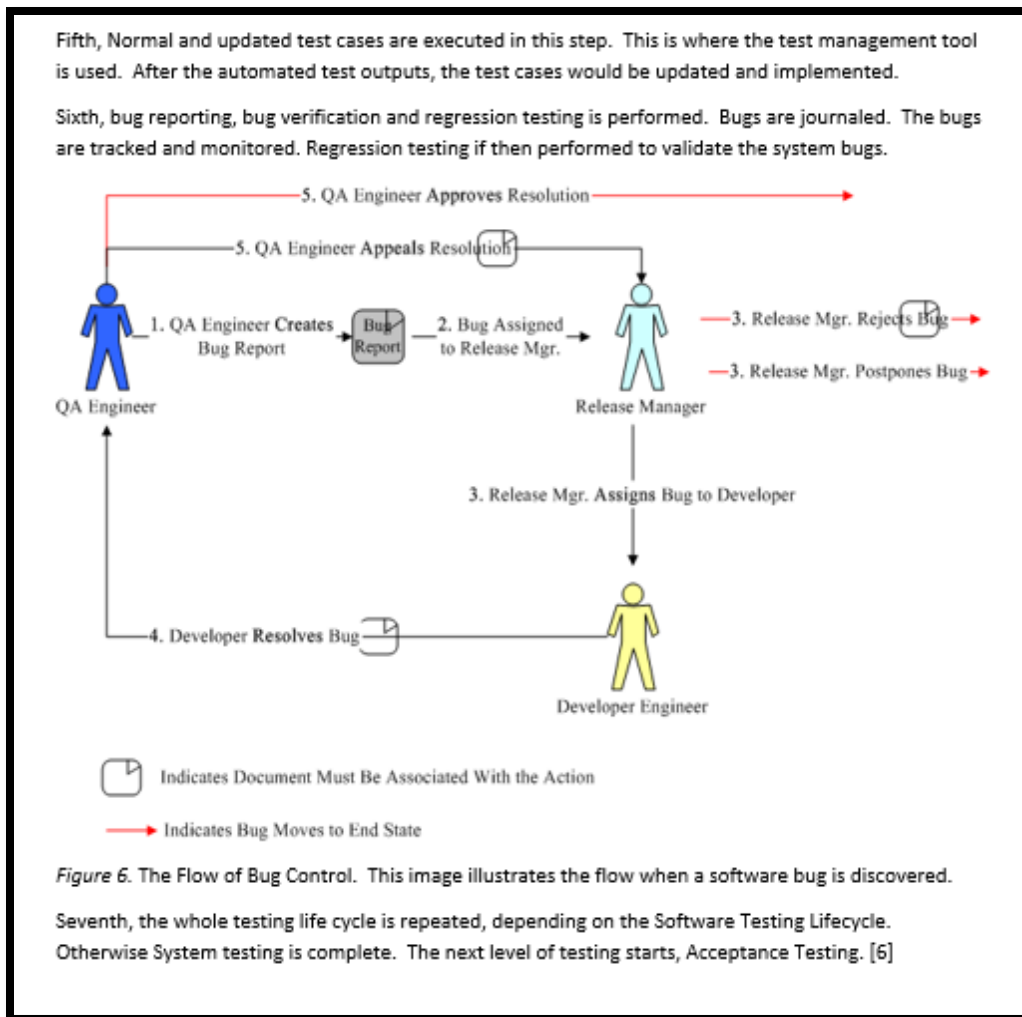


Figure 8. Continuation of the 5th page of System Testing LO

The sixth page of the System Testing Learning Objective is the reference list. This page lists all the references used to gather the information to create the learning objective.

References

[1] STF. (2016, August 7). System Testing [Web Tutorial Entry]. Retrieved from <http://softwaretestingfundamentals.com/system-testing/>

[2] Desikan, S. & Ramesh, G. (2012). System and Acceptance Testing. In H.N. Mahabala, S. Jose, & M.E. Sethurajan (Eds). Software Testing Principles and Practices (127-156). Noida, India: Dorling Kindersley Pvt. Ltd.

[3] Tutorialspoint.com. (2016, August 10). Software Testing – Levels [Web Tutorial Entry]. Retrieved from [http://www.tutorialspoint.com/software\\_testing/software\\_testing\\_levels.htm](http://www.tutorialspoint.com/software_testing/software_testing_levels.htm)

[4] Guru99. (2016, August 10). 100 Types of Testing You Never Knew Existed [Web Tutorial Entry]. Retrieved from <http://www.guru99.com/types-of-software-testing.html>

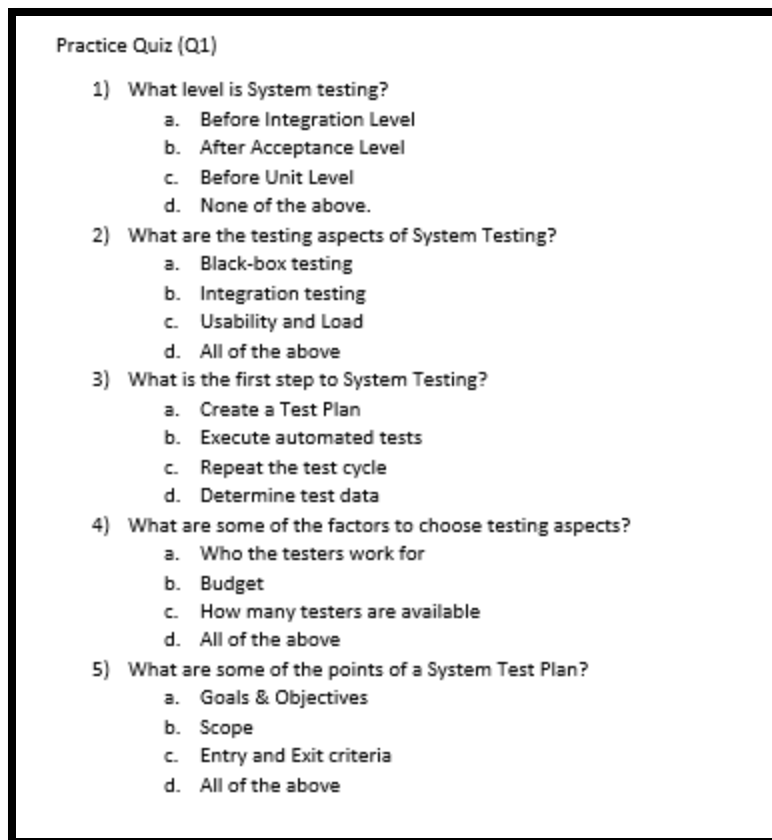
[5] Guru99. (2016, August 10). What is System Testing? [Web Tutorial Entry]. Retrieved from <http://www.guru99.com/system-testing.html>

[6] Softwaretestingclass.com. (2016, August 17). System Testing: What? Why? & How? [Web Tutorial Entry]. Retrieved from <http://www.softwaretestingclass.com/system-testing-what-why-how/>

<new page>

Figure 9. The 6th page of System Testing LO

The seventh page of the learning objective is the practice quiz. There are 10 multiple choice questions along with the correct answers. The main purpose of the practice quiz is to get the student ready for the real quiz that will be a part of their grade. The figure shows the practice quiz questions one through five along with the multiple-choice answers.



*Figure 10.* The 7th page of System Testing LO

Figure 11 illustrates multiple choice questions six through 10. It is a continuation of the practice quiz from figure 10. The figure also contains the correct answers to the quiz.

6) Is this level of testing used for debugging?

- a. Yes
- b. No

7) Who would perform System Testing?

- a. Release Team
- b. Quality Management Team
- c. Development Team
- d. A & B

8) What testing method is used for System Testing?

- a. White-box
- b. Unit
- c. Black-box
- d. None of the above

9) What are some of the items included on the Test Case?

- a. Test Scenario
- b. Test Entry Criteria
- c. Test Case ID
- d. A & C

10) What is the benefit of System Testing?

- a. The application is tested thoroughly to verify that it meets the functional and technical specifications.
- b. The bugs are identified and are corrected in this phase.
- c. System testing is less time consuming.
- d. A & C

Answers:1) d 2) c 3) a 4) d 5) d 6) b 7) d 8) c 9) c 10) a

<new page>

Figure 11. Continuation of the 7th page of System Testing LO

The eighth page of the learning objective is the Real Quiz. The quiz consists of 10 multiple choice questions including the correct answers. The purpose of the quiz is to determine the student's retention of the information. This will be a part of the final grading of the student. Figure 12 illustrates the 8th page of the System Testing Learning Objective. Figure 12 shows the Real Quiz, questions 1 through 5.

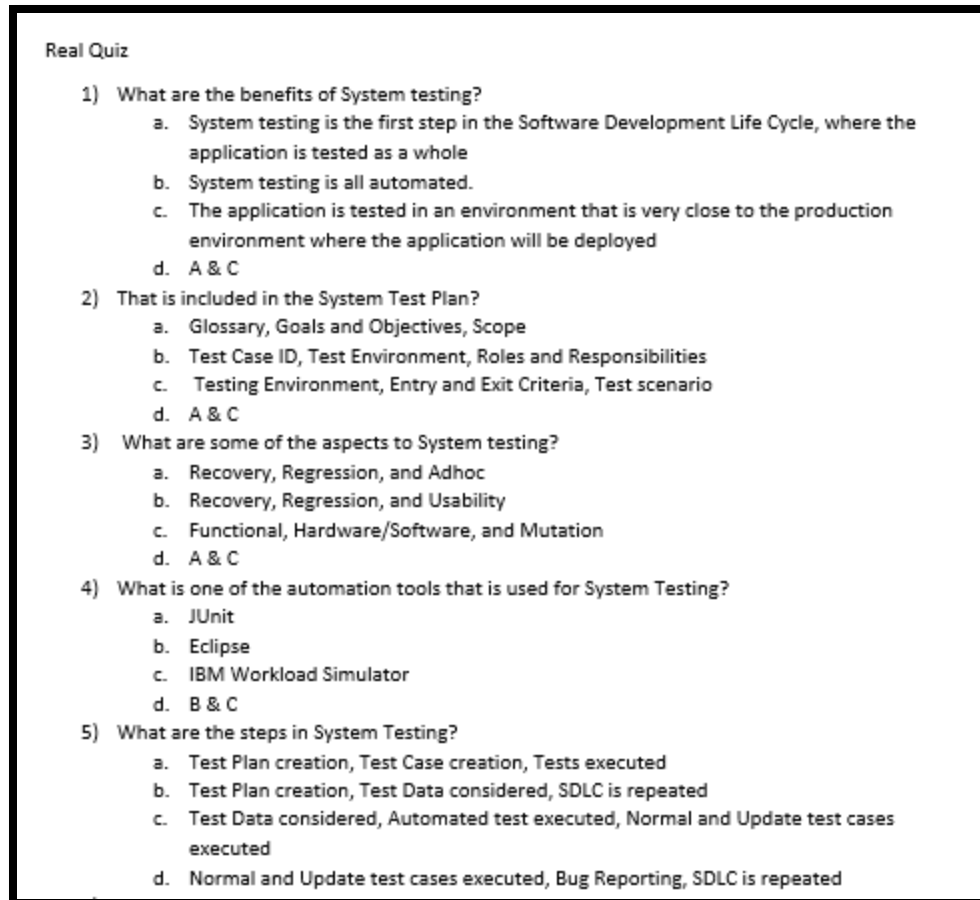


Figure 12. The 8th page of System Testing LO

Figure 13 lists the multiple-choice questions six through ten. It has the correct answers at the bottom of the page.

6) In the step of System Testing; bug reporting, bug verification and regression testing, is the bug fixed at that time?

- a. Yes
- b. No

7) Who performs System Testing?

- a. Development Team
- b. Project Managers
- c. Specialized Testing Team
- d. None of the above

8) What testing method is used for System Testing?

- a. White-box
- b. Grey-box
- c. Black-box
- d. B & C

9) When is System Testing performed?

- a. Before Unit Testing and after Integration Testing
- b. After Integration Testing and before Acceptance Testing
- c. After Unit Testing and before Integration Testing
- d. After Acceptance Testing and before Unit Testing

10) What are some of the items in the Test Case?

- a. Test Case ID, Test Suite Name, How to Test
- b. Test Data, Expected Result, Actual Result, and a Pass or Fail
- c. Test Case ID, Test Suite Name, Test Scenario Result
- d. A & B

Answers 1) d 2) a 3) b 4) c 5) c 6) b 7) c 8) c 9) b 10) d

Figure 13. Continuation of the 8th page of System Testing LO

### 3.3.2 Acceptance Testing LO

The Acceptance Testing LO includes a title, testing type, testing methods, tools used, basic concepts, and quizzes. Acceptance testing LO – LO01 is the title. This type of testing is mainly manual; consisting of checklists. The primary testing methods used are black-box, alpha, and beta. There are diverse ways to perform acceptance testing and different types of acceptance testing. For example, the type of software created plays a significant role in this. If the software is commercial-off-the-shelf, the acceptance testing method will be alpha and beta testing. If the software is custom-built, then it will be end-user in-house acceptance testing. The acceptance testing will include; Contract Acceptance Testing, Regulation Acceptance Testing, or Operational Acceptance Testing. [15] So, the whole point is to figure out what will benefit the student the most and review it. The students are mainly some type of IT background, therefore will be working more with custom-built software. The focus for this LO was on custom-built software. This was noted on the page. There are many tools used for this; eggplant, Ranorex 2, web2test, Zephyr, engageuat, and many more company driven tools (these are mainly used to report any bugs found, automated acceptance testing, and acceptance testing via internet). The main tools used are Excel spread sheets and checklists. Acceptance Testing basic concepts consist of; overview of concept, difference between System and Acceptance testing concepts, differences of COTS and Custom-Built software, UAT process, and the conclusion. The last items were the 2 quizzes. (Figures 10-13) These will determine the students' knowledge of the concept. The first page of Acceptance Testing Learning Objective is the main content page with the overview. At the top of the page is the title. Then there are the headers for the page implementation into WReSTT-CyLE website. The overview provides some of the background into acceptance testing and an analogy for better understanding.



## Acceptance Testing – LO01

### Content:

**Basic Concepts**

**Type: Manual**

**Method: Black-box, Alpha, Beta**

**Tools: eggPlant, Ranorex 2, web2test, Zephyr, engageuat, and many more company driven tools**

### Acceptance Testing:

Definition by ISTQB:

acceptance testing: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. [1]

Acceptance Testing (also called; User Acceptance Testing

(UAT), end-user testing, customer-driven testing, etc.) is the final level of testing.

This is where; the kind of software comes into consideration for testing. As

up to now in the testing life cycle, most of the testing is conducted the same way between the different kinds of software; commercial-off-the-shelf (COTS) or custom built. In Acceptance testing the software kind will help determine the aspects of testing used. The kind of software will determine these factors; the 'how', 'what', 'where', 'why' and 'who'.

Analog:

During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed. Once System Testing is complete, Acceptance Testing is performed so as to confirm that the ballpoint pen is ready to be made available to the end-users. [1]

<new page>

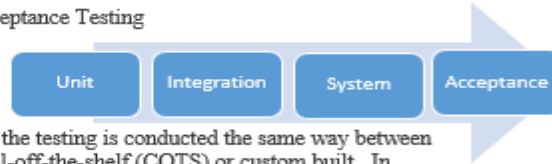


Figure 14. The 1st page of Acceptance Testing LO

The second page of the learning objective is the differences between System and Acceptance testing. The differences are formatted side by side for better viewing. There are eight items in the list, this listing was cut down because the other items found were like these listed.

<b>Difference in System Testing and Acceptance Testing?</b>	
<b>System Testing</b>	<b>Acceptance Testing</b>
1. System testing is end to end testing performed to check if the software meets the specified requirements.	1. Acceptance testing is functionality testing performed to check if the software meets the customer requirements.
2. System testing is performed by developers and testers.	2. Acceptance testing is performed by independent testers, stakeholders, and the clients.
3. System Testing can be both functional and nonfunctional testing	3. Acceptance testing is pure functional testing.
4. In this testing we test the software for complete specification including hardware and software, memory and number of users.	4. Here we test the software for the user needs and if user needs are met in the developed software.
5. It is performed with demo data and not the production data.	5. It is performed with actual real time data, production data.
6. System testing is performed before the Acceptance testing.	6. Acceptance testing is performed after the System testing.
7. System Testing comprises of System Testing and System Integration testing.	7. Acceptance testing comprises of alpha testing and beta testing.
8. System testing involves nonfunctional testing that is performance load and stress testing.	8. Acceptance testing involves functional testing that is boundary value analysis, equivalence partitioning and decision table testing.
<a href="http://www.softwaretestingclass.com/difference-between-system-testing-and-acceptance-testing/">http://www.softwaretestingclass.com/difference-between-system-testing-and-acceptance-testing/</a>	
<new page>	

Figure 15. The 2nd page of Acceptance Testing LO

The third page of the learning objective is the differences between COTS and Custom Built software. This topic was added because of the importance. The way the software is built determines which acceptance testing process used. If the software is COTS, then acceptance testing is done via Internet and the results captured. If the software is Custom Built, the acceptance testing is done on the customer site with real-time data.

**The Differences in COTS versus Custom Built Software in UAT**

Testing aspects: The COTS will mainly use Alpha and Beta testing aspects and custom will use manual-input and checklists. Both kinds of software will use automated testing and are Black Box.

Test cases/Test Plan: The test cases will relatively be the same. These are based on the requirements that are captured at the beginning of the software lifecycle. These test cases will already be implemented into the testing environment and will be automated. In the following example, the test case is about a login function for an application.

Test Case #	Test Case Description	Step #	Test Data	Steps to Perform	Expected Result	Actual Result	Comments
1-a	1. Login - with correct credentials	1	url: www.gmail.com	Launch URL	Gmail.com page should open up		
		2	Username: Test	Enter Username	The username is entered into the username field		
		3	Password: ALM Test	Enter Password	The password entered into the Password field and characters are hidden as dots		
		4		Click Sign In button	The user is logged into gmail.com page and inbox is displayed		

<http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2013/06/Quality-Center-Test-cases.jpg>

The assumption of Acceptance testing is that all of the other phases of testing are complete and there are no viable bugs in the system. For example—page does not load. Acceptance testing is not for ‘bug’ hunting, but more for performance testing for the user. For example; for a log-in user acceptance test: the test is looking for the timing that it takes to log-in; not that the user failed verification with the correct credentials. [2]

Location of testing: The assumption for this would be that the test cases are chosen and the test environment is implemented (if needed). The custom built software—the customer would be provided an area where the test would be performed. For COTS—the location of testing would be: a testing environment on site (Alpha Tests) or on the Internet (Beta Tests). [2] For the Beta testing, the test environment would be downloaded with the software, this would be error gathering software that would report back to the company.

Purpose of the tests: The purpose of UAT is either to determine if the customer will purchase the product or to decide if it is at a release state to end users called “go or no-go” decision. This decision is based on the results of the Acceptance test and the end user/customer/company. Here is

Figure 16. The 3rd page of Acceptance Testing LO

Figure 17 is a continuation of the third page of Acceptance Testing LO. It illustrates an example of a Go No Go checklist for testing.

an example of a company (New Flyer) Go/No Go checklist:

**New Flyer Go/No Go Checklist**

Item Number	Subject Area	Go-Live Criteria	Project Team Owner	Functional Owner	Condition Counterpart	QA Reviewer	Go/No Go Status (Red, Yellow, Green)	(To be Done, In Progress, Done)	Stop/Go Severity	Supplemental Documentation/Reference/Comment
1	Acceptance Test	Have tests been conducted against the production data?					Yellow		1	
2	Business Readiness	Has there been adequate communication to key personnel in the business unit (users, management, executives, suppliers, field personnel)?					Yellow		2	
3	Communications	Has the communication timeline for internal and external communication release been established?					Yellow		1	
4	Communications	Have the communication mechanisms (newletters, e-mail, etc.) been established?					Yellow		2	
5	Communications	Have the internal communication materials been created?					Yellow		2	
6	Communications	Have the communicators of the various communication items been identified?					Yellow		2	
7	Communications	Has the system availability during and after cutover been communicated to the application users?					Yellow		1	
8	Communications	Has the help desk process been communicated to the application users?					Yellow		1	
9	Communications	Have the contingency procedures and activating events been communicated to the application users?					Yellow		1	
10	Communications	Has the external communication to customers occurred?					Yellow		3	
11	Communications	Has the legacy system availability after conversion been communicated to the appropriate parties?					Yellow		2	
12	Communications	Has the final transition plan dates and impacts been communicated to executives?					Yellow		1	
13	Communications	Has the final transition plan been communicated to the parties involved in executing the cutover events?					Yellow		1	
14	Communications	Has the final transition plan dates and impacts been communicated to the application users?					Yellow		1	
15	Communications	Has the process for handling feedback and questions on deployed internal and external communication been defined and implemented?					Yellow		2	
16	Infrastructure	Have the data communication mechanisms been designed to be to handle system failure?					Yellow		1	
17	Business Readiness	Are users and management adequately trained on accessing and using system reports?					Yellow		2	
18	Training	Have the training audiences been identified?					Yellow		1	
19	Training	Have all users been given a training guide that explains new processes?					Yellow		1	

–<http://img.docstoccdn.com/thumb/orig/35216530.png>

For both, the bugs that are found will be captured and inventoried. These will either help determine; the go/no-go decision, to leave it as is, or to be fixed in an update at a later time. [3]

Testers: If the software is under ‘contract to hire’ then the customer/customer representative will perform the testing. If the software is not under contract, the software testing team, end users and/or internet users will perform the testing. [4] For custom built software, the company will produce the user manuals needed for the client software and will train the clients (that are performing the tests) before they perform the UAT.

<new page>

Figure 17. Continuation of the 3rd page of Acceptance Testing LO

The fourth page of the learning objective is the User Acceptance Testing (UAT) Process. This process is for custom built software. The relevance is that most students will be creating this type of software. There is a note on the top of the page informing that the rest of the learning objective will focus on custom built software user acceptance testing process. The figure illustrates steps 1, 2, and 3 of the UAT Process.

\*\*\* The Learning Object is focused on custom-built software from this point onward. \*\*\*

**UAT Process:**

Before UAT is started, there are some requirements that need to be completed. All other testing is complete and at a passing level. The business requirements are available to the UAT testing team. The UAT environment is up and functional and there is documentation signing off that it is ready with the correct testing committee members. All high-priority or showstopper bugs are fixed and verified and all regression testing completed. Also, the sign-off from the System testing team that their work is complete is needed. Only when these are completed UAT can be started. [4][5]

1. Analyse the business requirements. This will be performed by the UAT Testing committee. The UAT committee can be comprised of; some of the testing team members, a business analyst, customer representative, and/or client appointed end users. In this step; Service Level Agreements (SLA) might become a part of the acceptance criteria. These are from the contract that initiated the project. For example; downtime of the system is less than 0.1%. So how to validate these in the UAT, this is done by having the right information and resources available to the client. This can be in the training of the employees or fully document any trouble shooting. [6]
2. Create a UAT test plan. This will be completed by the testing team or business analyst. Get plan. For an example of a test plan please click here: [hot ref to the test plan](#) (can add this to the website in the references-this is a great template) [7]
3. Identify the UAT Test Scenarios. The UAT Testing committee will choose the requirements that best fit the criteria for accepting the application. [8] Some of the acceptance testing criteria would be;

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance
- Timeliness
- Confidentiality and Availability
- Installability and Upgradability
- Scalability
- Documentation

Figure 18. The 4th page of Acceptance Testing LO

Figure 19 is a continuation of the fourth page of Acceptance Testing LO. It includes steps 4,5, and 6 of the UAT process.

4. Create UAT Test Cases. This is performed by the UAT Testing Team. These are very detailed in most cases. The entry and exit data is given along with the all the steps in between. Most of these will be already in the testing system and have been tested numerous times through-out the software life cycle. Before the final UAT is completed, the testing team will take the finalized test cases and run them in-house. This will be automated testing.

5. Execute the UAT Test Cases. This will be completed on-site of the client company. The testers will be picked by the UAT Testing committee. This will be manual testing. In some cases, the test data can be 'live' data from the client. [9] These will be performed by the testers in-front of the Release Team and UAT Testing committee. In some cases, 'live' data will be used for the inputs. In these cases, it is best to scramble the data for security reasons and train the user on the data flow (if needed). [9]

6. Confirm the business objectives. The UAT Testing committee will confirm the results with the UAT Testing committee and sign-off. This is the final step of validating the business requirements within the software. [5] UAT is complete. The companies will have the final meeting and decide on the go/no-go decision at this point.

<new page>

Figure 19. Continuation of 4th page of Acceptance Testing LO

The fifth page of the learning objective is the conclusion. This page gives a brief overview of the information that was presented. It stresses the importance of how the software is built in the UAT process.

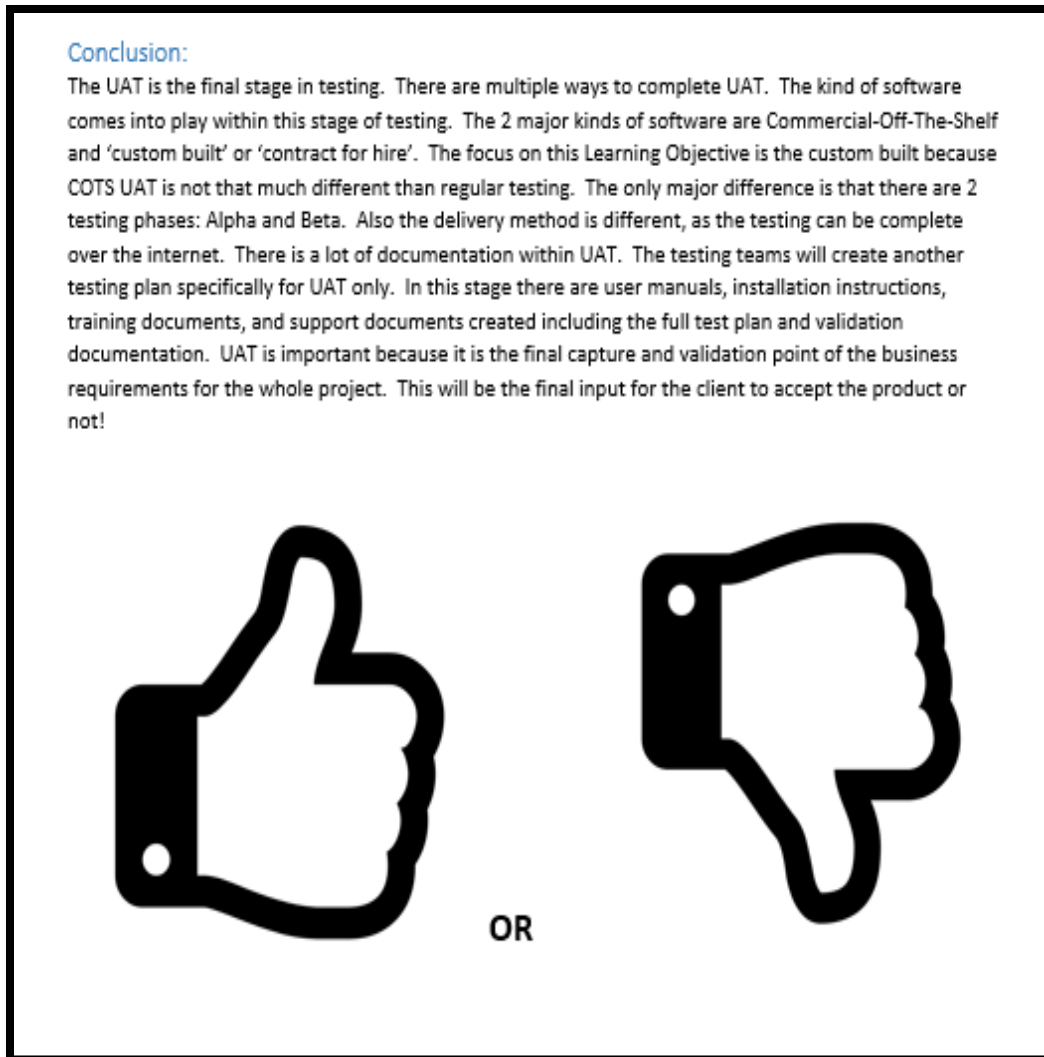


Figure 20. The 5th page of Acceptance Testing LO

The sixth page of the learning objective is the Practice Quiz. There are 10 multiple choice questions along with the correct answers. The main purpose of the practice quiz is to get the student ready for the real quiz that will be a part of their grade. Figure 21 illustrates the practice quiz along with the correct answers.

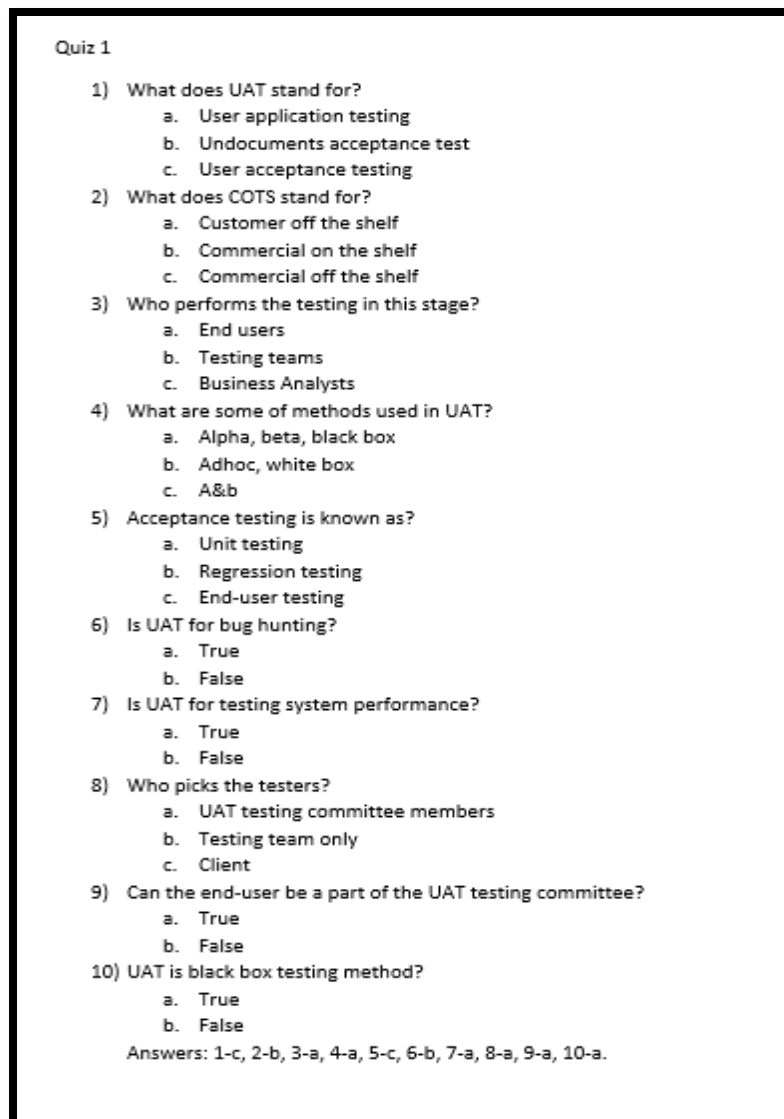


Figure 21. The 6th page of Acceptance Testing LO



The seventh page of the learning objective is the Real Quiz. The quiz consists of 10 multiple choice questions including the correct answers. The purpose of the quiz is to determine the student's retention of the information. This will be a part of the final grading of the student. Figure 22 illustrates the real quiz along with correct answers.

Quiz 2

- 1) What is the main focus of this LO?
  - a. System Testing
  - b. Custom built software
  - c. Software Release Team
- 2) What are some of the testing criteria?
  - a. Documentation, Usability, Data integrity
  - b. Usability, affordability, maintainability
  - c. Sunny-day, rainy-day, grey-day
- 3) What does SLA stand for?
  - a. Software learning assessment
  - b. Service level agreement
  - c. Service level agent
- 4) Who performs the testing in this stage?
  - a. End uses
  - b. Testing team
  - c. UAT testing committee
- 5) What are some of the documents created in this stage?
  - a. Training documents
  - b. Warranty information
  - c. None of the above
- 6) Who is included in the UAT testing committee?
  - a. Product company CEO
  - b. Subject knowledge expert
  - c. End-user, testing team, business analysts
- 7) UAT is not the final stage in testing?
  - a. True
  - b. False
- 8) Are UAT test cases automated?
  - a. True
  - b. False
- 9) Is regression testing perform in this stage?
  - a. True
  - b. False
- 10) What is another name for 'contract for hire'?
  - a. UAT
  - b. COTS
  - c. Custom-built

Answers: 1-b, 2-a, 3-b, 4-a, 5-a, 6-c, 7-b, 8-a, 9-a, 10-c.

Figure 22. The 7th page of Acceptance Testing LO

The final page of the learning objective is the references. This page lists all of the references used in the learning objective.

**References:**

[1] STF. (2016, September 10). Acceptance Testing Fundamentals. [Web Log Page]. Retrieved from <http://softwaretestingfundamentals.com/acceptance-testing/>

[2] Techopedia.com. (2016, September 12). User Acceptance Testing (UAT). [Web Definition Entry]. Retrieved from <https://www.techopedia.com/definition/3887/user-acceptance-testing-uat>

[3] S Seela. (2016, August 26). What is User Acceptance Testing (UAT) and How to Perform It Effectively? [Web Article Entry]. Retrieved from <http://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/>

[4] Software Testing Class. (2012, October 7). User Acceptance Testing: What? Why? & How? [Web Article Entry]. Retrieved from <http://www.softwaretestingclass.com/user-acceptance-testing-what-why-how/>

[5] Software Testing Class. (2012, October 7). Difference between System testing and Acceptance Testing. [Web Article Entry]. Retrieved from <http://www.softwaretestingclass.com/difference-between-system-testing-and-acceptance-testing/>

[6] Desikan, S, & Ramesh, G. (2012). Software Testing: Principles and Practices. *System and Acceptance Testing*. (pp. 158-162). New Dehli, India: Pearson.

[7] Coley Consulting. (2016, September 15). UAT Test Plan Template. [Web Log Entry]. Retrieved from [http://coleyconsulting.co.uk/UAT\\_Test\\_Plan.htm](http://coleyconsulting.co.uk/UAT_Test_Plan.htm)

[8] TutorialsPoint. (2016, September 20). What is Acceptance Testing? [Web Article Entry]. Retrieved from [http://www.tutorialspoint.com/software\\_testing\\_dictionary/acceptance\\_testing.htm](http://www.tutorialspoint.com/software_testing_dictionary/acceptance_testing.htm)

[9] Guru99. (2016, September 26). What is User Acceptance Testing (UAT)? [Web Log Entry]. Retrieved from <http://www.guru99.com/user-acceptance-testing.html>

Figure 23. The 8th page of Acceptance Testing LO

## 4. FUTURE WORK

There is more work to be done with this topic. There is a lot of information that can go more in-depth on the subject. For example, in the System Testing – LO; 02 and 03 can be added. There are some concepts not on the site yet, these can all be added. Some more tutorials on different testing tools can be added, there are many different tools available that were not at the creation of this site.

The current WReSTT-CyLE has not been studied since the new design. This can be testing within a software or testing undergraduate course and journaled. Some studies can be on how effective are the LO and quizzes.

There can be more options added to the site itself. An example is the option of Instructor uploading their own LO or tutorials. Another option could be Instructor requests on a specific topic or tool. Within WReSTT-CyLE, the website can be opened to the public for use. As mentioned, not all IT professionals go through the traditional route of schooling. This can also help current professionals with up-to-date tutorials and LOs for ramping up on a specific tool or topic.

## 5. DISCUSSION AND CONCLUSION

There is a gap of testing skills/knowledge in IT professionals. The way to address this issue is to teach students the importance of testing and provide them with the proper tools to gain this knowledge. Academia has started to resolve this with WReSTT-CyLE. It is a collaborative, interactive cyber-learning environment for testing concepts. It is minimally intrusive to any classroom. All of the data and learning items are accessed via internet. The site was updated with gamification and visual components to motivate students to participate within the site. The WReSTT-CyLE provides learning objectives for testing concepts. These learning objectives support the instructors and teach the students in a very logical way. Along with the gamification and easy access for the students, the site is very motivational for a learning environment. Recent studies reported that the site supports this very effort.

## REFERENCES

1. Software Testing Class Team. (2016, September 30). Importance of Testing. [Web Log Entry]. Retrieved from <http://www.softwaretestingclass.com/importance-of-testing/>
2. Kovach, S. (2016 October 11). It's time for Samsung to come clean about the Galaxy Note 7. Business Insider: Tech Insider. Retrieved from <http://www.businessinsider.com/samsung-galaxy-note-7-recall-what-went-wrong-2016-10>
3. Software Testing Class Editors. (2012, October 7). Importance of Testing. [Web Article Entry]. Retrieved from <http://www.softwaretestingclass.com/>
4. Lent, J. (2016, September 15). QA skills gap: Testing pros need enough to wrote a test script. Retrieved from <http://searchsoftwarequality.techtarget.com/feature/QA-skills-gap-Testing-pros-need-enough-to-write-a-test-script>
5. Guru99. (2016, August 10). Software Testing as a career- complete guide. Retrieved from <http://www.guru99.com/>
6. Goswami, A., Walia, G. S., & Abufardeh, S. (2014). Using a Web-Based Testing Tool Repository in Programming Course: An Empirical Study. Proceedings of 2014 International Conference on Frontiers in Education: Computer Science and Computer Engineering. July 21-24, FECS 2014 USA.
7. WReSTT-CyLE, (2016, September 25). Home Page. Retrieved from <http://demo.wrestt.cis.fiu.edu/about-wrestt-com>
8. Fu, Y., & Clarke, P.J. (2016). Gamification based Cyber Enabled Learning Environment of Software Testing. 123rd ASEE Annual Conference & Exposition. Accepted 03/28/16
9. Clarke, P.J. (2009). Introductions to Web-Based Repository for Software Testing Tools (WReSTT). [PPT Format]. Retrieved from <http://demo.wrestt.cis.fiu.edu/wistpc-09>

10. Clarke, P.J., Pava, J., Wu, Y. & King, T. M. (2011). Collaborative Web-Based Learning of Testing Tools in SE Courses. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (2011). ACM, New York, NY, USA, pages 147-152.
11. Fu, Y., Barnes, N., & Clarke, P.J. (2016). Integrating Software Testing into Computer Science Curriculum Using WReSTT-CyLE. 123rd ASEE Annual Conference & Exposition.
12. Gamification. (n.d.) In Merriam-Webster dictionary online. Retrieved from <http://www.merriam-webster.com/dictionary/gamification>
13. Hamari, J., Koivisto, J. & Sarsa, H. (2014). Does Gamification Work? — A Literature Review of Empirical Studies on Gamification. In Proceedings of the 47<sup>th</sup> IEEE Hawaii International Conference on System Science. HICSS 2014, pages 3025-3034.
14. ISTQB Team. (2016, September 15). What is Acceptance testing? [Web Log Entry]. Retrieved from <http://istqbexamcertification.com>
15. Peham, T. (2015). 5 Types of user acceptance tests – the perfect UAT framework. [Web Log Entry]. Retrieved from <http://usersnap.com/blog/types-user-acceptance-tests-frameworks>
16. Wiley, David A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D.A. Wiley (Ed.). The Instructional Use of Learning Objects
17. Martinez, M. (2000). Designing learning objects to mass customize and personalize learning. In D.A. Wiley (ED.) The Instructional Use of Learning Objects.
18. Mills, Sandy. (2002). Learning about Learning Objects with Learning Objects [on-line]. Available: [http://www.alivetek.com/learningobjects/site\\_paper.htm](http://www.alivetek.com/learningobjects/site_paper.htm).
19. Softwaretestingclass.com. (2016, August 17). System Testing What? Why? & How? [Web Tutorial Entry}. Retrieved from <http://www.softwaretestingclass.com/system-testing-what-why-how/>

## APPENDIX A. LISTING OF BOOKS REVIEWED

Desikan, S., & Ramesh, G. (2006). *Software Testing: Principles and Practices*. Pearson Education India.

Kit, E., & Finzi, S. (1995). *Software testing in the real world: Improving the process*. New York, NY: ACM Press.

Mathur, A. (2013). *Foundations of software testing*. Delhi: Pearson.

Pezzè, M., & Young, M. (2008). *Software testing and analysis: Process, principles, and techniques*. Hoboken, NJ: Wiley.

Tsui, F. F., & Karam, O. (2007). *Essentials of software engineering*. Sudbury, MA: Jones and Bartlett.

Wieggers, K. E. (2003). *Software requirements: Practical techniques for gathering and managing requirements throughout the product development cycle*. Redmond, WA: Microsoft Press.

## APPENDIX B. A LISTING OF WEBSITES VISITED

Coley Consulting. (2016, September 15). UAT Test Plan Template. [Web Log Entry]. Retrieved from [http://coleyconsulting.co.uk/UAT\\_Test\\_Plan.htm](http://coleyconsulting.co.uk/UAT_Test_Plan.htm)

etestinghub, (2016, September 17). Online software testing tutorial - manual and automation. Retrieved from <http://www.etestinghub.com/>

Gamification. (n.d.) In Merriam-Webster dictionary online. Retrieved from <http://www.merriam-webster.com/dictionary/gamification>

Guru99. (2016, August 10). Software Testing as a career- complete guide, What is System Testing?, What is User Acceptance Testing (UAT)?, & 100 Types of Testing You Never Knew Existed [Web Tutorial Entry]. Retrieved from <http://www.guru99.com/>

Hower, R., (2016, September 12) What are some recent major computer system failures caused by software bugs? (n.d.). Retrieved from <http://www.softwareqatest.com/qatfaq1.html>

ISTQB Team. (2016, September 15). What is Acceptance testing? & Why is software testing necessary? [Web Log Entry]. Retrieved from <http://istqbexamcertification.com/why-is-testing-necessary/>

Kovach, S. (2016 October 11). It's time for Samsung to come clean about the Galaxy Note 7. Business Insider: Tech Insider. Retrieved from <http://www.businessinsider.com/samsung-galaxy-note-7-recall-what-went-wrong-2016-10>

Lent, J., (2016, September 15). QA skills gap: Testing pros need enough to write a test script. Retrieved from <http://searchsoftwarequality.techtarget.com/feature/QA-skills-gap-Testing-pros-need-enough-to-write-a-test-script>

Peham, T., (2015). 5 Types of user acceptance tests – the perfect UAT framework. Retrieved from <http://usersnap.com/blog/types-user-acceptance-tests-frameworks>



Seela, S. (2016, August 26). What is User Acceptance Testing (UAT) and How to Perform It Effectively? [Web Article Entry]. Retrieved from <http://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/>

Software Testing Class Editors. (2012, October 7). Difference between System testing and Acceptance Testing, Importance of Testing, System Testing: What? Why? & How?, & User Acceptance Testing: What? Why? & How?. [Web Article Entry]. Retrieved from <http://www.softwaretestingclass.com/>

Software Testing Fundamentals. (2016, September 10). Acceptance Testing Fundamentals & System Testing. [Web Log Page]. Retrieved from <http://softwaretestingfundamentals.com/>

Software Testing Help. (2016, August 24). System Testing – A Beginner’s Guide. [Web Article Entry]. Retrieved from <http://www.softwaretestinghelp.com/system-testing/>

Techopedia.com. (2016, September 12). User Acceptance Testing (UAT). [Web Definition Entry]. Retrieved from <https://www.techopedia.com/definition/3887/user-acceptance-testing-uat>

Thomson, R., (2008, May 8) Lack of software testing to blame for Terminal 5 fiasco, BA executive tells MPs. (n.d.). Retrieved from <http://www.computerweekly.com/news/2240085948/Lack-of-software-testing-to-blame-for-Terminal-5-fiasco-BA-executive-tells-MPs>

TutorialsPoint. (2016, August 10). Software Testing – Levels &What is Acceptance Testing? [Web Tutorial Entry]. Retrieved from [http://www.tutorialspoint.com/software\\_testing/](http://www.tutorialspoint.com/software_testing/)

WReSTT-CyLE, (2016, September 25). Home Page, About, Instructor Home Page, Student, Home Page, Events, Forums, Documentation, Software Testing Tools Tutorials, & Software Testing Learning Objectives. Retrieved from <http://demo.wrestt.cis.fiu.edu/about-wrestt-com>

Online Presentation Slides:

Clarke, P.J. (2009). Introductions to Web-Based Repository for Software Testing Tools (WReSTT).

[PPT Format]. Retrieved from <http://demo.wrestt.cis.fiu.edu/wistpc-09>

## APPENDIX C. A LISTING OF CONFERENCE PAPERS REVIEWED

- Clarke, P.J., Allen, A. A., King, T. M., Jones, E. L., & Natesan, P. (2010). Using a Web-Based Repository to Integrate Testing Tools into Programming Courses. In Proceedings of the ACM international Conference Companion on Object Oriented Programming Systems Languages and Applications Companion. SPLASH 2010, pages 193-200.
- Clarke, P.J., Davis, D. L., Chang-lau, R. & King, T. M. (2014). Observations on Student Use of Tools in an Undergraduate Testing Class. In Proceedings of the 121st American Society for Engineering Education (ASEE) - Software Engineering Constituent Committee Division Track (SWECC) 2014. Paper ID: 10123, 16 pages.
- Clarke, P.J., Davis, D. L., King, T. M., Pava, J., & Jones, E. L. (2014). Integrating Testing into Software Engineering Courses Supported by a Collaborative Learning Environment. ACM Trans. Comput. Educ. 14, 3, Article 18 (October 2014), 33 pages.
- Clarke, P.J., Pava, J., Davis, D. L., & King, T. M. (2012). Using WReSTT in SE Courses: An Empirical Study. In the Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12). ACM, pages 307-312.
- Clarke, P.J., Pava, J., Wu, Y. & King, T. M. (2011) Collaborative Web-Based Learning of Testing Tools in SE Courses. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11). ACM, New York, NY, USA, pages 147-152.
- Fu, Y., Barnes, N., & Clarke, P.J. (2016). Integrating Software Testing into Computer Science Curriculum Using WReSTT-CyLE. 123rd ASEE Annual Conference & Exposition. Accepted 03/28/16
- Fu, Y., & Clarke, P.J. (2016). Gamification based Cyber Enabled Learning Environment of Software Testing. 123rd ASEE Annual Conference & Exposition. Accepted 03/28/16

Goswami, A., Walia, G. S., & Abufardeh, S. (2014). Using a Web-Based Testing Tool Repository in Programming Course: An Empirical Study. Proceedings of 2014 International Conference on Frontiers in Education: Computer Science and Computer Engineering. July 21- 24, FECS 2014 USA.

## APPENDIX D. A LISTING OF ALL MATERIAL REVIEWED

1. Acceptance Testing: Formal testing conducted to determine whether a system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system. It is usually performed by the customer.
2. Accessibility Testing: Type of testing which determines the usability of a product to the people having disabilities (deaf, blind, mentally disabled etc.). The evaluation process is conducted by persons having disabilities.
3. Active Testing: Type of testing consisting in introducing test data and analysing the execution results. It is usually conducted by the testing teams.
4. Agile Testing: Software testing practice that follows the principles of the agile manifesto, emphasizing testing from the perspective of customers who will utilize the system. It is usually performed by the QA teams.
5. Age Testing: Type of testing which evaluates a system's ability to perform in the future. The evaluation process is conducted by testing teams.
6. Ad-hoc Testing: Testing performed without planning and documentation - the tester tries to 'break' the system by randomly trying the system's functionality. It is performed by the testing teams.
7. Alpha Testing: Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end user.
8. Assertion Testing: Type of testing consisting in verifying if the conditions confirm the product requirements. It is performed by the testing teams.
9. API Testing: Testing technique similar to unit testing in that it targets the code level. API Testing differs from unit testing in that it is typically a QA task and not a developer task.
10. All-pairs Testing: Combinatorial testing method that tests all possible discrete combinations of input parameters. It is performed by the testing teams.

11. Automated Testing: Testing technique that uses automation testing tools to control the environment set-up, test execution and results reporting. It is performed by a computer and is used inside the testing teams.
12. Basis Path Testing: A testing mechanism which derives a logical complexity measure of a procedural design and use this as a guide for defining a basic set of execution paths. It is used by testing teams when defining test cases.
13. Backward Compatibility Testing: Testing method which verifies the behaviour of the developed software with older versions of the test environment. It is performed by testing teams.
14. Beta Testing: Final testing before releasing application for commercial purpose. It is typically done by end-users or others.
15. Benchmark Testing: Testing technique that uses representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration. It is performed by testing teams.
16. Big Bang Integration Testing: Testing technique which integrates individual program modules only when everything is ready. It is performed by the testing teams.
17. Binary Portability Testing: Technique that tests an executable application for portability across system platforms and environments, usually for conformation to an ABI specification. It is performed by the testing teams.
18. Boundary Value Testing: Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams.
19. Bottom Up Integration Testing: In bottom up integration testing, module at the lowest level are developed first and other modules which go towards the 'main' program are integrated and tested one at a time. It is usually performed by the testing teams.

20. Branch Testing: Testing technique in which all branches in the program source code are tested at least once. This is done by the developer.
21. Breadth Testing: A test suite that exercises the full functionality of a product but does not test features in detail. It is performed by testing teams.
22. Black box Testing: A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality. It is performed by QA teams.
23. Code-driven Testing: Testing technique that uses testing frameworks (such as xUnit) that allow the execution of unit tests to determine whether various sections of the code are acting as expected under various circumstances. It is performed by the development teams.
24. Compatibility Testing: Testing technique that validates how well a software performs in a particular hardware/software/operating system/network environment. It is performed by the testing teams.
25. Comparison Testing: Testing technique which compares the product strengths and weaknesses with previous versions or other similar products. Can be performed by tester, developers, product managers or product owners.
26. Component Testing: Testing technique similar to unit testing but with a higher level of integration - testing is done in the context of the application instead of just directly testing a specific method. Can be performed by testing or development teams.
27. Configuration Testing: Testing technique which determines minimal and optimal configuration of hardware and software, and the effect of adding or modifying resources such as memory, disk drives and CPU. Usually it is performed by the performance testing engineers.

28. Condition Coverage Testing: Type of software testing where each condition is executed by making it true and false, in each of the ways at least once. It is typically made by the automation testing teams.
29. Compliance Testing: Type of testing which checks whether the system was developed in accordance with standards, procedures and guidelines. It is usually performed by external companies which offer "Certified OGC Compliant" brand.
30. Concurrency Testing: Multi-user testing geared towards determining the effects of accessing the same application code, module or database records. It is usually done by performance engineers.
31. Conformance Testing: The process of testing that an implementation conforms to the specification on which it is based. It is usually performed by testing teams.
32. Context Driven Testing: An Agile Testing technique that advocates continuous and creative evaluation of testing opportunities considering the potential information revealed and the value of that information to the organization at a specific moment. It is usually performed by Agile testing teams.
33. Conversion Testing: Testing of programs or procedures used to convert data from existing systems for use in replacement systems. It is usually performed by the QA teams.
34. Decision Coverage Testing: Type of software testing where each condition/decision is executed by setting it on true/false. It is typically made by the automation testing teams.
35. Destructive Testing: Type of testing in which the tests are carried out to the specimen's failure, in order to understand a specimen's structural performance or material behaviour under different loads. It is usually performed by QA teams.
36. Dependency Testing: Testing type which examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality. It is usually performed by testing teams.



37. **Dynamic Testing:** Term used in software engineering to describe the testing of the dynamic behaviour of code. It is typically performed by testing teams.
38. **Domain Testing:** White box testing technique which contains checking that the program accepts only valid input. It is usually done by software development teams and occasionally by automation testing teams.
39. **Error-Handling Testing:** Software testing type which determines the ability of the system to properly process erroneous transactions. It is usually performed by the testing teams.
40. **End-to-end Testing:** Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate. It is performed by QA teams.
41. **Endurance Testing:** Type of testing which checks for memory leaks or other problems that may occur with prolonged execution. It is usually performed by performance engineers.
42. **Exploratory Testing:** Black box testing technique performed without planning and documentation. It is usually performed by manual testers.
43. **Equivalence Partitioning Testing:** Software testing technique that divides the input data of a software unit into partitions of data from which test cases can be derived. It is usually performed by the QA teams.
44. **Fault injection Testing:** Element of a comprehensive test strategy that enables the tester to concentrate on the way the application under test can handle exceptions. It is performed by QA teams.
45. **Formal verification Testing:** The act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. It is usually performed by QA teams.

46. Functional Testing: Type of black box testing that bases its test cases on the specifications of the software component under test. It is performed by testing teams.
47. Fuzz Testing: Software testing technique that provides invalid, unexpected, or random data to the inputs of a program - a special area of mutation testing. Fuzz testing is performed by testing teams.
48. Gorilla Testing: Software testing technique which focuses on heavily testing of one particular module. It is performed by quality assurance teams, usually when running full testing.
49. Gray Box Testing: A combination of Black Box and White Box testing methodologies: testing a piece of software against its specification but using some knowledge of its internal workings. It can be performed by either development or testing teams.
50. Glass box Testing: Similar to white box testing, based on knowledge of the internal logic of an application's code. It is performed by development teams.
51. GUI software Testing: The process of testing a product that uses a graphical user interface, to ensure it meets its written specifications. This is normally done by the testing teams.
52. Globalization Testing: Testing method that checks proper functionality of the product with any of the culture/locale settings using every type of international input possible. It is performed by the testing team.
53. Hybrid Integration Testing: Testing technique which combines top-down and bottom-up integration techniques in order leverage benefits of these kind of testing. It is usually performed by the testing teams.
54. Integration Testing: The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.
55. Interface Testing: Testing conducted to evaluate whether systems or components pass data and control correctly to one another. It is usually performed by both testing and development teams.

56. Install/uninstall Testing: Quality assurance work that focuses on what customers will need to do to install and set up the new software successfully. It may involve full, partial or upgrades install/uninstall processes and is typically done by the software testing engineer in conjunction with the configuration manager.

57. Internationalization Testing: The process which ensures that product's functionality is not broken and all the messages are properly externalized when used in different languages and locale. It is usually performed by the testing teams.

58. Inter-Systems Testing: Testing technique that focuses on testing the application to ensure that interconnection between application functions correctly. It is usually done by the testing teams.

59. Keyword-driven Testing: Also, known as table-driven testing or action-word testing, is a software testing methodology for automated testing that separates the test creation process into two distinct stages: a Planning Stage and an Implementation Stage. It can be used by either manual or automation testing teams.

60. Load Testing: Testing technique that puts demand on a system or device and measures its response. It is usually conducted by the performance engineers.

61. Localization Testing: Part of software testing process focused on adapting a globalized application to a particular culture/locale. It is normally done by the testing teams.

62. Loop Testing: A white box testing technique that exercises program loops. It is performed by the development teams.

63. Manual Scripted Testing: Testing method in which the test cases are designed and reviewed by the team before executing it. It is done by manual testing teams.

64. Manual-Support Testing: Testing technique that involves testing of all the functions performed by the people while preparing the data and using these data from automated system. It is conducted by testing teams.

65. Model-Based Testing: The application of Model based design for designing and executing the necessary artefacts to perform software testing. It is usually performed by testing teams.
66. Mutation Testing: Method of software testing which involves modifying programs' source code or byte code in small ways in order to test sections of the code that are seldom or never accessed during normal tests execution. It is normally conducted by testers.
67. Modularity-driven Testing: Software testing technique which requires the creation of small, independent scripts that represent modules, sections, and functions of the application under test. It is usually performed by the testing team.
68. Non-functional Testing: Testing technique which focuses on testing of a software application for its non-functional requirements. Can be conducted by the performance engineers or by manual testing teams.
69. Negative Testing: Also known as "test to fail" - testing method where the tests' aim is showing that a component or system does not work. It is performed by manual or automation testers.
70. Operational Testing: Testing technique conducted to evaluate a system or component in its operational environment. Usually it is performed by testing teams.
71. Orthogonal array Testing: Systematic, statistical way of testing which can be applied in user interface testing, system testing, regression testing, configuration testing and performance testing. It is performed by the testing team.
72. Pair Testing: Software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyses or reviews the testing. This can be done between one Tester and Developer or Business Analyst or between two testers with both participants taking turns at driving the keyboard.
73. Passive Testing: Testing technique consisting in monitoring the results of a running system without introducing any special test data. It is performed by the testing team.

74. Parallel Testing: Testing technique which has the purpose to ensure that a new application which has replaced its older version has been installed and is running correctly. It is conducted by the testing team.

75. Path Testing: Typical white box testing which has the goal to satisfy coverage criteria for each logical path through the program. It is usually performed by the development team.

76. Penetration Testing: Testing method which evaluates the security of a computer system or network by simulating an attack from a malicious source. Usually they are conducted by specialized penetration testing companies.

77. Performance Testing: Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

78. Positive Testing: Testing process where the system validated against the valid input data. In this testing, tester always checks for only valid set of values and checks if an application behaves as expected with its expected inputs.

79. Qualification Testing: Testing against the specifications of the previous release, usually conducted by the developer for the consumer, to demonstrate that the software meets its specified requirements.

80. Ramp Testing: Type of testing consisting in raising an input signal continuously until the system breaks down. It may be conducted by the testing team or the performance engineer.

81. Regression Testing: Type of software testing that seeks to uncover software errors after changes to the program (e.g. bug fixes or new functionality) have been made, by retesting the program. It is performed by the testing teams.

82. Recovery Testing: Testing technique which evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. It is performed by the testing teams.

83. Requirements Testing: Testing technique which validates that the requirements are correct, complete, unambiguous, and logically consistent and allows designing a necessary and sufficient set of test cases from those requirements. It is performed by QA teams.

84. Security Testing: A process to determine that an information system protects data and maintains functionality as intended. It can be performed by testing teams or by specialized security-testing companies.

85. Sanity Testing: Testing technique which determines if a new software version is performing well enough to accept it for a major testing effort. It is performed by the testing teams.

86. Scenario Testing: Testing activity that uses scenarios based on a hypothetical story to help a person think through a complex problem or system for a testing environment. It is performed by the testing teams.

87. Scalability Testing: Part of the battery of non-functional tests which tests a software application for measuring its capability to scale up - be it the user load supported, the number of transactions, the data volume etc. It is conducted by the performance engineer.

88. Statement Testing: White box testing which satisfies the criterion that each statement in a program is executed at least once during program testing. It is usually performed by the development team.

89. Static Testing: A form of software testing where the software isn't used it checks mainly for the sanity of the code, algorithm, or document. It is used by the developer who wrote the code.

90. Stability Testing: Testing technique which attempts to determine if an application will crash. It is usually conducted by the performance engineer.

91. Smoke Testing: Testing technique which examines all the basic components of a software system to ensure that they work properly. Typically, smoke testing is conducted by the testing team, immediately after a software build is made.

92. Storage Testing: Testing type that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. It is usually performed by the testing team.
93. Stress Testing: Testing technique which evaluates a system or component at or beyond the limits of its specified requirements. It is usually conducted by the performance engineer.
94. Structural Testing: White box testing technique which takes into account the internal structure of a system or component and ensures that each program statement performs its intended function. It is usually performed by the software developers.
95. System Testing: The process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It is conducted by the testing teams in both development and target environment.
96. System integration Testing: Testing process that exercises a software system's coexistence with others. It is usually performed by the testing teams.
97. Top Down Integration Testing: Testing technique that involves starting at the stop of a system hierarchy at the user interface and using stubs to test from the top down until the entire system has been implemented. It is conducted by the testing teams.
98. Thread Testing: A variation of top-down testing technique where the progressive integration of components follows the implementation of subsets of the requirements. It is usually performed by the testing teams.
99. Upgrade Testing: Testing technique that verifies if assets created with older versions can be used properly and that user's learning is not challenged. It is performed by the testing teams.
100. Unit Testing: Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

101. User Interface Testing: Type of testing which is performed to check how user-friendly the application is. It is performed by testing teams.

102. Usability Testing: Testing technique which verifies the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. It is usually performed by end users.

103. Volume Testing: Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner. It is usually conducted by the performance engineer.

104. Vulnerability Testing: Type of testing which regards application security and has the purpose to prevent problems which may affect the application integrity and stability. It can be performed by the internal testing teams or outsourced to specialized companies.

105. White box Testing: Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers.

106. Workflow Testing: Scripted end-to-end testing technique which duplicates specific workflows which are expected to be utilized by the end-user. It is usually conducted by testing teams.