

CHAPTER 1

INTRODUCTION TO TESTING

“Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements, or to identify differences between expected and actual results”

- Institute of Electrical and Electronics Engineers (IEEE).

Software applications have grown in recent years from simple console based applications that perform a single task. Now they are complex, visually pleasingly and animated programs that perform a variety of tasks. It is no longer unusual for a single application to perform complex tasks, integrate with many other applications as well as multiple back-end systems, and process documents in a variety of formats. An example of this is online banking.

The features available in these software applications have substantially advanced out of a need to support a more sophisticated and computer savvy user base. Additionally, today’s users have many more choices of computing devices such as PCs, laptops, PDAs, Tablet PCs, Smartphones, Pocket PCs etc. Also a great variety of operating system choices are available for these devices and all of these systems require software to be developed for performing various tasks.

The need for testing these software applications is therefore very clear because testing helps to ensure that applications developed meet with the original intent of the customer. The complexity that applications must support today comes at a price to developers because each of the many features being created have to be tested in multiple scenarios and different environments to ensure that they work properly as intended. And in the event that the user performs an invalid operation, the system should generate a meaningful message to the user about why the failure occurred.

It is not unusual to download and use an application and find that an attempt to do something impossible, like save a document directly to a CDROM drive, will cause the application to crash. This is because the developer of the application did not foresee the multiple scenarios that end-users of the application would attempt while using the application. While the end-user might be wrong to save a document to a read-only drive, it is still important for the application to recover from the scenario gracefully without causing the user to lose the work done to their document.

This is precisely why we do software testing. It provides the ability to rigorously test a software application to ensure that the application meets with the originating requirements and expectations. Testing also helps to ensure that the application continues to function even when used in unintended ways. If problems are found during the development phase of an application, they can be fixed before the application is made available to end-users.

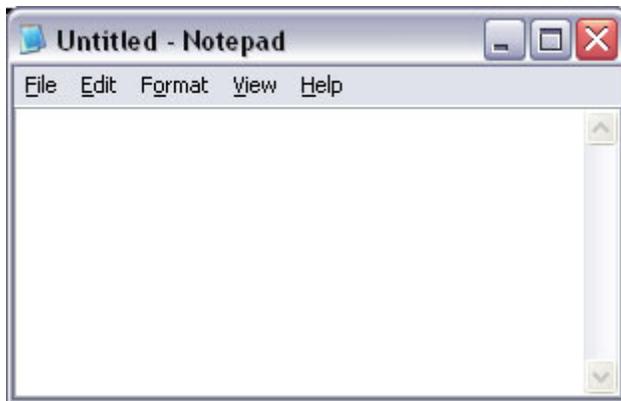


Figure 1.1: The Windows Notepad application.

This directly impacts the cost of development. You may be familiar with the 1:10:100 rule of testing that states that the cost to fix a defect increases exponentially the longer it takes to identify the defect. Testing thereby helps ensure that the end-user is provided with a quality product and an error-free experience.

An example of the exponential growth in application features can be seen in document editing software. Notepad, a standard application shipped with all versions of Microsoft Windows and shown in Figure 1.1, is a simple text editor capable of editing text documents.

NOTE: The notepad application is accessible from Start ► Programs ► Accessories ► Notepad

Long since the standard for quick document editing exercises, Notepad is rapidly losing its dominant place to a host of new tools that provide more advanced features that are useful while editing text documents. Textpad (www.textpad.com) shown in Figure 1.2 is an example of the new generation of text editors. The versatile document editing application that can be used to edit documents of many formats providing such useful features as formatting, multiple document view and syntax highlighting.

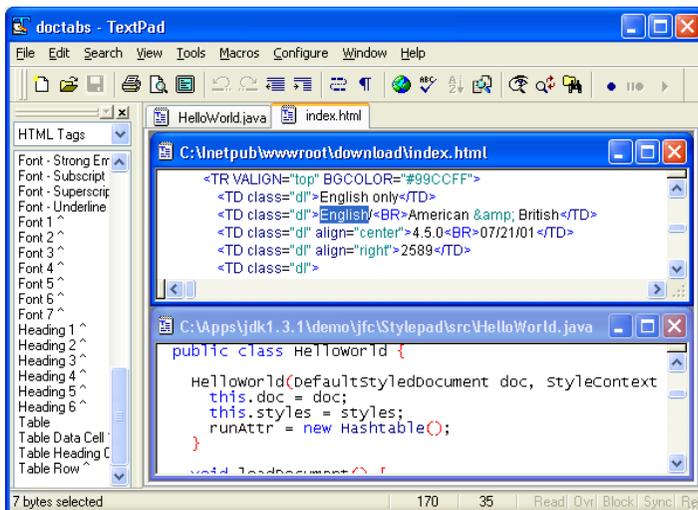


Figure 1.2: The Textpad application

In Figure 1.2 Textpad is shown editing both a HTML and Java document in two windows. Notice that it provides syntax highlighting features to these different file formats.

Developers of editing tools such as this would need to put the application through a rigorous process to test all these advanced features.

Software Development Life Cycle

The development of software applications is based on a lifecycle process known as the software development lifecycle (SDLC). SDLC is a software engineering process that provides information about steps that need to be taken to advance a software project from concept to deployment. SDLC is broken into 5 phases with each phase dependent upon results from the preceding phase. Each phase also specifies what must happen during that portion of the software process. The testing process, which we are primarily concerned with in this book, is the fourth phase in the SDLC process.

SDLC helps describe an approach which, when followed properly, ensures that applications meet both the functional and visual requirements stipulated in their creation documents. The stages contained within the SDLC process include:

Requirements Analysis – The gathering of details about what needs to be done, and how the application should perform those tasks. This phase is handled by Business Analysts (BA) who must meet with all the stakeholders in the application to capture all the needs and details of what they want the application to do. The most difficult part of this process is ensuring that the requirements are:

Complete – containing all necessary functionalities needed by the users.

Non-conflicting – Often with a large user base it is easy to have a group of users need a feature that conflicts with features needed by other users. This must be negotiated during this stage because the application cannot be built with conflicting requirements.

Unambiguous – written in clear language so that the same meaning can be derived when different parties read the document.

The result of this process is determining the features that must exist in the final application.

Design Phase – High-level design of the application to specify how the application will be created. This is often managed by a software architecture team who take into consideration the programming language and software platform in which the application will be developed. These details will determine how the internal structure of the application will be designed. Additional activities during this phase includes the creation of flowcharts that describe how components within the application will interact, designing the screen layout that dictate what the application would look like, and also creating the database structures that the application might use to store data.

Implementation and Unit Testing – Involves writing the code for the application using the chosen programming language and software platform from the design phase. A software application being created in the Java programming for instance might be coded using the Abstract Windowing Toolkit (AWT), or Java Foundation Classes (JFC) or even the Standard Widget Toolkit (SWT) for the object library. The choice of the tool to use is made during the design phase, and the implementation would simply create the application conforming to this decision. The choice of interface technology determines the actual type of software code that the developer will write. During this process, unit testing is performed at the code level to ensure the correctness of the software functions that are created by the programmer. Unit testing also ensures the several functions can work together in completing tasks.

Testing – The testing phase covers the process of planning tests that will be run on the application to examine its functionality. The phase also describes how the test would be executed. Decisions are made about using a manual or automated approach and test cases are created based on the chosen test strategy. During this phase, the goal is to test the entire application to ensure that when all the separate components that make up the application are assembled, the application functions as described in the requirements documents.

Maintenance – On completion and release of an application, there is often a need for corrective maintenance to handle defects that were undiscovered in the development process, and preventive maintenance to mitigate issues that might arise from external threats. Additionally there is often a desire to upgrade certain functionality for the application. For deployed applications such as your bank’s web application, there is also a need to support users and manage configurations of the software system. The maintenance cycle describes how this process should be handled.

From the above, you may notice that the SDLC process is originally intended to be a linear process with one phase completing before the commencement of the next. You must have requirements before you know how to design your database structure, and likewise you need this database structure to know what tables and fields to reference in your programming code.

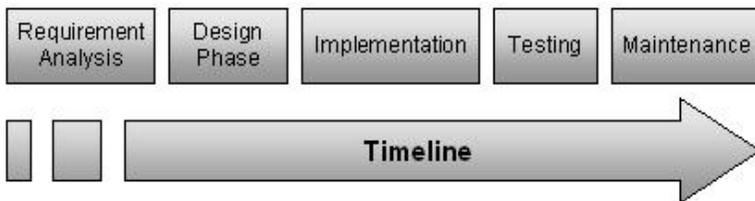


Figure 1.3: SDLC Timeline

Figure 1.3 shows the SDLC process implemented linearly. However, as time is always a major factor in software development with many companies seeking to reduce the time-to-

market for new products, methodologies have been created to optimize SDLC by allowing certain parts of it to function in parallel. These methodologies are processes designed to maximize the efficiency of the SDLC. Some of these methodologies include Scrum, Agile, Extreme Programming (XP) and the Rational Unified Process (RUP). The underlying goal of each these methodologies are to optimize the software development timeline while ensuring the quality of the resulting product.

By staggering the starting time of the subsequent phases, it is often possible to achieve a level of parallel development. As shown in Figure 1.4, this has a net effect of saving the project time by having multiple things done simultaneously. Note that the difference in the methodology being used will impact how the phases are staggered.

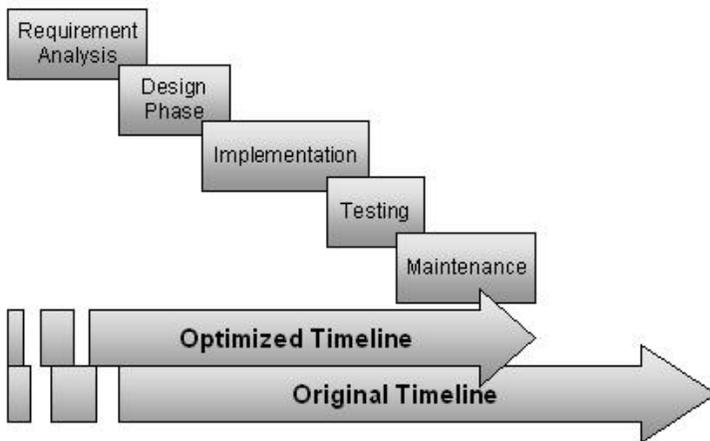


Figure 1.4: Optimized SDLC Timeline showing concurrent phases.

ASK THE EXPERTS

Q: Should the functional tester understand the programming language the application was created in?

A: This is not necessary, during functional testing we are only exposed to the completed application and not to the source code. Functional testing ignores the internal mechanisms of an application and focuses on the output provided for a given input.

Q: What is the best software development methodology?

A: This question is analogous to asking “what is the best programming language?” Each methodology has its strong suit and expectedly its weaknesses. All of them however, seek to make it easier for a project to be completed successfully on-schedule, and on-budget.

SDLC Testing Phase

The testing phase as a part of the SDLC process governs the preparation for testing, the management of testing resources and the creation of test cases. It also manages the execution of these tests as well the creation and tracking of defects found within the software application.

Although there is less consensus about the phases of the software testing process than there is about the SDLC process, you can generally find that the testing software systems is broken down in to the following phases:

Test Planning - The goal of this phase is to create a process that specifies the details as well as the logistics involved with the testing process. It takes into account the tools that will be used to test the application, the testing environment etc. Usually, a test plan document results from this phase, describing in details such crucial items as:

- Roles and responsibilities of everyone involved in the testing process
- Process of setting up the test environment
- Obtaining and managing test data
- Standards for test creation and execution
- Preliminary schedule for testing
- Test management process
- The defect-tracking process
- Software tools that would be used through the testing process

Test Design - Involves the breakdown of the functional and performance requirements for the application into a series of test cases. These test cases are documents that describe how a specific functionality is going to be tested. In order to successfully test any functionality, it is important that the test case completely identifies what needs to be done. Also, the test case needs to specify the sequential process that needs to be followed, any results expected and include all other relevant pieces of information. Table 1.1 contains a list of useful information that would be found in a test case.

Item	Description
Test Name	The name that would be given to the test script. This would allow a correlation between the test case that describes what is being tested and the test script that performs the test.
Purpose	The feature that is being tested.
Pre-Condition	The initial condition that must be met for the feature being tested to be properly tested.
User action	A sequential step-wise listing of actions that the user is expected to perform in order to appropriately test the desired functionality.
Expected result of each action	The result that is expected after each action performed by the user.
Post-Condition	The condition in which the test environment should be returned to on the completion of this test.
Valid Test Data	Source and location where valid test data can be found to test the functionality being tested. This data will be either hard-coded into the test script, or preferably, entered into the data table and parameterized into the test

Possible Exceptions	List of possible exceptions that can occur during this test. This information is used for creating recovery scenarios.
---------------------	------------------------------------------------------------------------------------------------------------------------

Table 1.1: Useful data needed in a test case

We have provided below in Listing 1.1, a test case that contains all this useful information. You should note that using tools such as Mercury Interactive’s Test Director, it is easy to capture and track this crucial information.

Test Name:
Add Expense

Purpose:
Verify that when expense records are added they display within the application and are added to the database

Pre-Condition:
Application is launched, with the main page displayed.
Note the number of expense records displayed

Step	User Action	Expected Result
1	Click the Today's Date button	Correct system date appears in the available textbox
2	Enter in additional details for an expense record	Entered details are displayed
3	Click the save button	a. Expense is displayed in the grid b. Totals' textbox is update c. Expense record is saved to database

Post-Condition:
Close the application

Valid Test Data:

Category: Movies

Description: LOTR Return of the King at Lennox

Amount: 11.50

Possible Exceptions:

N/A

Listing 1.1: Add Expense Test Case

Test Development - This phase is only necessary for software applications that will be tested using automated software testing tools. In this phase, the test cases created in the previous stage, are converted into executable scripts. These scripts will then be executed using an automated testing tool. This step is unnecessary for applications that are being manually tested because manual testing involves the performance of the contents in the test case by individuals.

We will provide more discussion on manual vs. automated testing in the next chapter.

Summary

In this chapter you have learned:

What software testing is, and why we test software applications.

The software development lifecycle, and where testing fits into this process.

How to shorten the software development lifecycle by making certain phases run concurrently.

The different parts of the software testing process.