International Journal of Advanced Science and Technology
Vol. 49, December, 2012

Test Cases Design for Software Database Provisioning Development

Sunguk Lee

Research Institute of Industrial Science and Technology
Pohang, Gyeongbuk, South Korea
sunguk@rist.re.kr

Abstract

This paper presents the concepts of test case based approach for the development of software systems. A variety of software development approaches are presented wherein they share some common methodology stages wherein testing plays a vital role for the success of the product. Software testing provides an objective, independent view of the software to allow the end-user to appreciate and understand the risks of software implementation. Examples of test cases for developed software have been presented.

Keywords: *Test Case, Software Development, Software Testing*

1. Introduction

The fast growing market demand for innovative software solutions motivates the software development technology to evolve into a more defined, managed and delivered. The drive for the need of a better quality control of the software development process has given rise to the discipline of software engineering that aims to apply the systematic approach exemplified in the engineering paradigm to the process of software development. Significant advantages and disadvantages are present to the various software development methodologies and approaches, but the best approach to solving a problem with software will often depend on the type of problem. Choosing the best approach may vary depending on the analysis of the problem at hand.

Among these approaches, software testing plays a significant role for the success of the implementation of the software solution. It is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. It is provided to ensure that the software product:

- meets the requirements specified (e.g. the user expects),
- desired functions work as expected,

This paper presents the concepts of test cases design for the development of software systems. There are a variety of software development approaches and they share some common methodology stages wherein testing plays a vital role for the success of the product. Software testing provides an objective, independent view of the software to allow the end-user to appreciate and understand the risks of software implementation.

The rest of this paper is organized as follows: Section 2 discusses about the different software development approaches; Section 3 outlines the concepts of a Test Case and Examples for Test Cases are designed; and the concluding remarks in Section 4.

2. Software Development Approaches

Software development refers to as the development of a software product that includes activities of computer programming. It is also known as application development, software design, designing software, software application development, enterprise application development, or platform development. It may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products [1].

Different approaches can be used for a software development; some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. The following are some of the stages or phases that most methodologies have in common: [1]:

- Analysis Phase
- Requirements Phase
- Design Phase
- Implementation (coding) Phase
- Testing
- Deployment
- Maintenance and bug fixing

These stages are often referred to collectively as the software development lifecycle, or SDLC. Different approaches to software development may carry out these stages in different orders, or devote more or less time to different stages. Significant advantages and disadvantages are present to the various methodologies, but the best approach to solving a problem with software will often depend on the type of problem. Choosing the best approach may vary depending on the analysis of the problem at hand [1].

2.1 Waterfall based Approach

The Waterfall model is a traditional sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance [1, 2].

The basic principles are [2]:

- Project is divided into sequential phases, with some overlap and splashback acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

2.2 Prototyping

Software prototyping is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of the software program being developed [1, 2].

The basic principles are [2]:

- Not a standalone, complete development methodology, but rather an approach to handle selected parts of a larger, more traditional development methodology (i.e. incremental, spiral, or rapid application development (RAD)).
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- User is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation.
- Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
- While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.
- A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.

2.3 Spiral Approach

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. It is a meta-model, a model that can be used by other models [2].

The basic principles are:

- Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
- "Each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program." Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration.
- Begin each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment

2.4 RAD Approach

Rapid application development (RAD) is a software development methodology, which involves iterative development and the construction of prototypes [2].

The basic principles are:

- Key objective is for fast development and delivery of a high quality system at a relatively low investment cost.
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- Key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
- Active user involvement is imperative.
- Iteratively produces production software, as opposed to a throwaway prototype.
- Produces documentation necessary to facilitate future development and maintenance.
- Standard systems analysis and design methods can be fitted into this framework.

2.5 Other Approaches

Other methodology practices include [2]:

- Object-oriented development methodologies known as object-oriented analysis and design (OOAD) [4].
- Top-down programming.
- Unified Process (UP) is an iterative software development methodology framework, based on Unified Modeling Language (UML). One popular version of UP is the Rational Unified Process (RUP).
- Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve via collaboration between self-organizing cross-functional teams.

3. Test Cases Design for Software Development

Test cases in software development refer to the set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites [3, 5].

A single test case is usually a single step, or occasionally a sequence of steps, to test the correct behaviour/functionalities, features of an application. An expected result or expected outcome is usually given [3].

Information that is included within a test case:

- test case ID
- test case description
- test step or order of execution number
- related requirement(s)

- depth
- test category
- author
- check boxes for whether the test is automatable and has been automated.
- Expected Result and Actual Result.
- Additional fields that may be included and completed when the tests are executed:
- pass/fail
- remarks

Larger test cases may also contain prerequisite states or steps, and descriptions. Figure 1 shows an example flow of events in software (e.g. point of sale system). The following example is included in the testing phase of the development of a Point of Sale System (POS). As shown in Figure 1, the flow of steps for processing a sale is presented. Table 1 depicts an example of a Test Case for a normal processing of a sale. Table 2 and Table 3 shows sample Test Cases for not correct functionality as the software is being tested. Expected results are shown to determine whether the test pass or fails.

Table 1. Test Case for Processing Sale

Test Case Name:	**Process Sale - Normal**
Use Case Name:	Process Sale
Use Case Path to be exercised:	[U1, S1, U2.1, S2.1, U3.1, S3.1, U4.1, S4.1]
Input Data:	12345
Initial Condition:	Cashier (Benjo) is allowed to process a sale transaction No item is in the shopping cart New sale transaction is already created Item 12345 is a valid input
Expected Result:	New sale transaction is still open for additional items Cashier is still authorized to work on the current transaction. 1 line item is now on the shopping cart New line item has been established for item 12345. System is ready to accept for additional item.

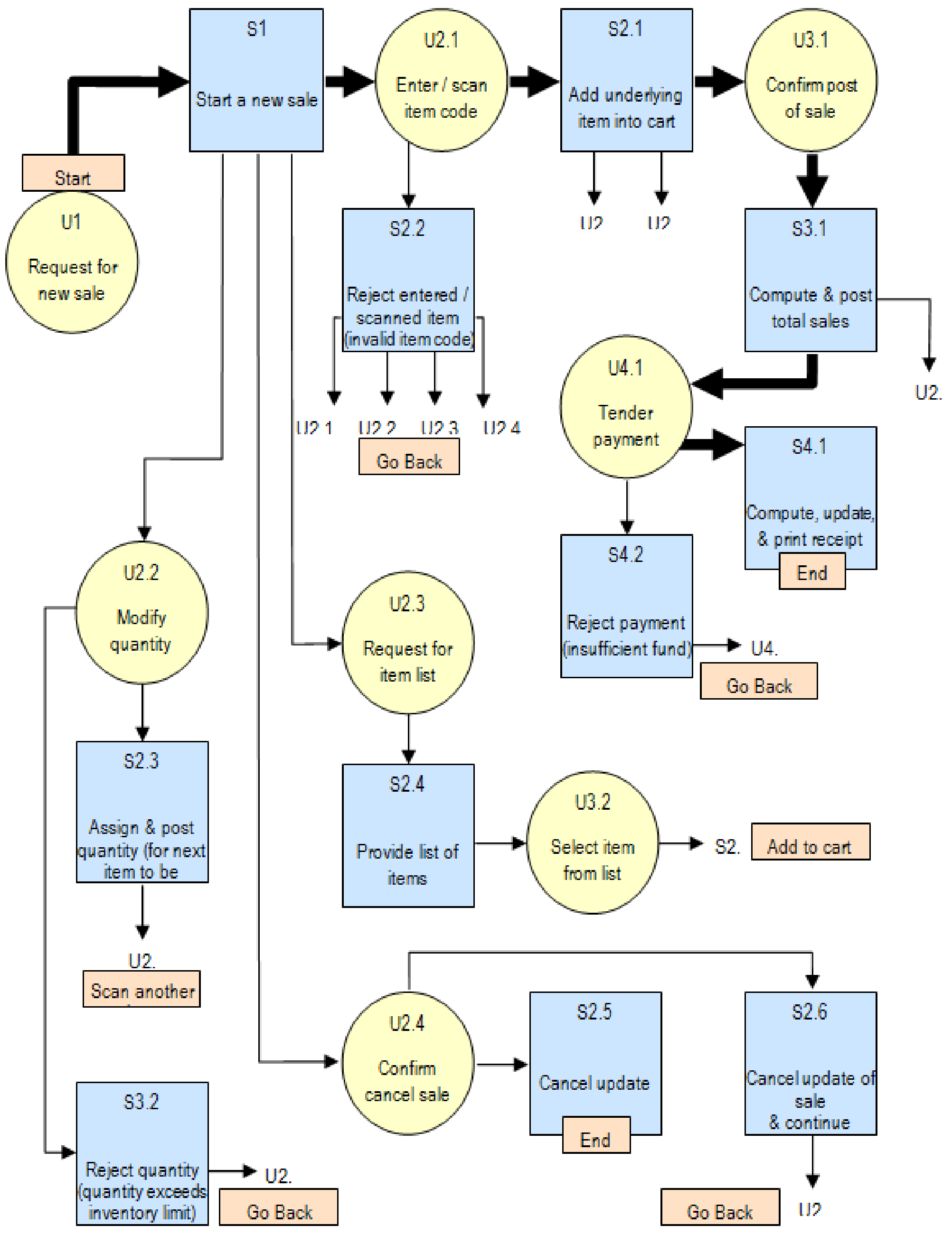

Figure 1. Flow of Processing Sale Event of a Point of Sale (POS)

Table 2. Test Case for Processing Sale - Negative

Test Case Name:	**Process Sale - item is not registered/recognized**
Use Case Name:	Process Sale
Use Case Path to be exercised:	[U1, S1, U2.1, S2.2]
Input Data:	54321
Initial Condition:	Cashier (Benjo) is allowed to process a sale transaction No item is in the shopping cart New sale transaction is already created Item 54321 is not a registered or recognized item
Expected Result:	New sale transaction is still open for additional items Cashier is still authorized to work on the current transaction. Still there are no items added in the shopping cart No new line has been established. Displays an error message for unregistered or unrecognized item System is ready to accept for additional item.

Table 3. Test Case for Processing Sale - Negative

Test Case Name:	**Process Sale - quantity exceeds inventory limit**
Use Case Name:	Process Sale
Use Case Path to be exercised:	[U1, S1, U2.2, S3.2]
Input Data:	7
Initial Condition:	Cashier (Benjo) is allowed to process a sale transaction New sale transaction is already created 1 item is in the shopping cart Default value of item's quantity is 1 Inventory limit for the item in the cart is 5 7 is a valid input
Expected Result:	New sale transaction is still open for additional items Cashier is still authorized to work on the current transaction. Still there are no items added in the shopping cart No new line has been established. Displays an error message for item that exceeds inventory limit System is ready to accept for additional item.

Figure 2 shows the flow of managing items event for a Point of Sale (POS) system. The flow of activities will show the behaviour of the functionality of the system as it will be tested. Table 4 depicts the normal behaviour of a Test Case for managing items while Table 5 presents a Test Case for managing items with insufficient item details supplied and duplicated item codes.

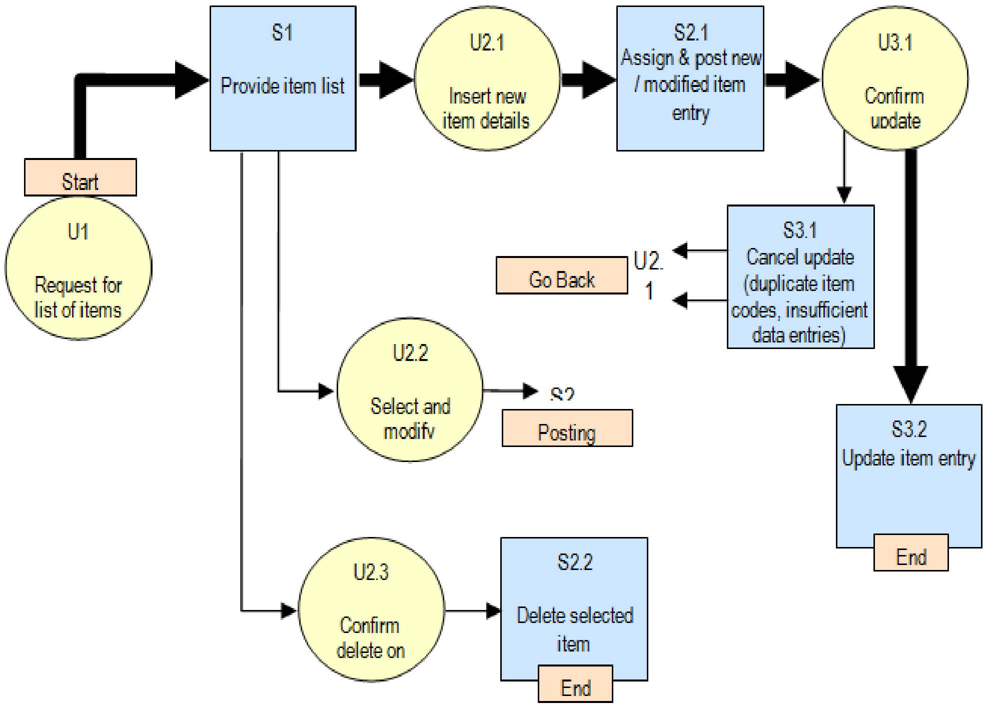

Figure 2. Flow of Managing Items Event of a Point of Sale (POS)

Table 4. Test Case for Managing Items

Test Case Name:	**Manage Items - Normal**
Use Case Name:	Manage Items
Use Case Path to be exercised:	[U1, S1, U2.1, S2.1, U3.1, S3.1, U3.1, 3.2]
Input Data:	67890-Bear Brand-Nestle-Milk-Lynlyn Poultry Supply-20.00-22.00-10
Initial Condition:	Only the administrator is allowed to manage items There are existing number of items in the database Add New Item transaction is active No duplicate of item code
Expected Result:	Admin is still authorized to work on another new transaction. New record has been appended on the database for item 67890. The form is ready to receive a new item as another entry.

Table 5. Test Case for Managing Items - Negative

Test Case Name:	**Manage Items - insufficient item details supplied/ duplicate item codes**
Use Case Name:	Manage Items
Use Case Path to be exercised:	[U1, S1, U2.1, S2.1, U3.1, S3.1, U3.1, S3.1]
Input Data:	67890-Nido-Nestle-Milk-Lynlyn Poultry Supply-20.00-22.00-10
Initial Condition:	Only the administrator is allowed to manage items There are existing number of items in the database The admin has already selected a type of item to be added Add New Item transaction is active Item code 67890 is already a duplicate of another item in the database
Expected Result:	New sale transaction is still open for additional items Admin is still authorized to work on the current transaction. No record is appended/updated in the database System is ready to accept for another item System is ready to accept additional lacking details

Software's complexity and accelerated development schedules make avoiding defects difficult. To assure quality software, the defects found during the testing and implementation phase will be given solutions or at least minimize the occurrence. The summary of defects must then be summarized and identify necessary actions to address and correct these defects as shown in Table 6.

Table 1. Summary of Defects

Defects	**Resolutions**
Process Sale – item is not registered/recognized	Displays an error message for unregistered or unrecognized item
Process Sale – quantity exceeds inventory limit	
Manage Items – insufficient item details supplied/ duplicate item codes	
----	---

4. Conclusions

Software testing refers to an investigation conducted to provide assurance of the quality of the software product or service under test. It includes processes of executing a program or application with the intent of finding software bugs (errors or other defects).

It is provided to ensure that the software product:

- meets the requirements specified (e.g. the user expects),
- desired functions work as expected.

This paper presents the concepts of test case based approach for the development of software systems. Test cases for software testing have been designed to determine whether an application or software system is working correctly or not

References

[1] http://en.wikipedia.org/wiki/Software_development.
[2] http://en.wikipedia.org/wiki/Software_development_methodology.
[3] http://en.wikipedia.org/wiki/Test_case.
[4] Georges Gauthier Merx & Ronald J. Norman (2006). Unified Software Engineering with Java. p.201.
[5] http://en.wikipedia.org/wiki/Oracle_(software_testing).