A Software Support Tool for Teaching Software Inspection

NASIM GAZERANI

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

MARCH, 2011
A Software Support Tool for Teaching Software Inspection

NASIM GAZERANI

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY UNIVERSITY OF MALAYA KUALA LUMPUR

MARCH, 2011
UNIVERSITI of MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: NASIM GAZERANI (I.C/Passport No: D14821901)

Registration/Matric No: WGC080010

Name of Degree: Master of Software Engineering


Field of Study: Requirement Engineering

I do solemnly and sincerely declare that:

1) I am the sole author/writer of this Work;
2) This Work is original;
3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya (“UM”), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate’s Signature Date: 2.3.2011

Subscribed and solemnly declared before,

Witness’s Signature Date:

Name:

Designation:
Abstract

The objective of inspection process is to reduce the cost by finding and removing defects earlier. In recent years, there have been a number of attempts to further increase inspection efficiency by the introduction of tool support, resulting in a number of prototype systems. Moreover, many software engineers suffer from lack of background knowledge and experience on software inspections and its techniques. The purpose of the study is to investigate the possible ways to educate Software engineers with high efficiency and to develop a teaching tool with appropriate features. So, we offered a teaching framework to make software engineering students have a deep understanding on software inspection and to improve their practical abilities. The framework consists of three parts: general guidelines, specific guidelines and learning activity. In addition, there are two parts of general guidelines: Software inspection concept and Software inspection technique. Specific guidelines include Software inspection process and technique processing of Software inspection. The third layer includes conceptual samples, collaboration in practice, work sample and assessment for improving practical skills and abilities. Based on the teaching framework, a software support is designed and developed by interactive feature such as combining text, sound, graphic, images and animation. The tool is evaluated using software engineering students in University of Malaya to shows its usefulness and effectiveness. We found that the tool is useful and effectiveness to be used as a teaching tool.
ACKNOWLEDGEMENTS

These two years of research on my Master thesis have been a thoroughly enjoyable experience, which I have already begun to look back at with nostalgia. I would like to take this opportunity to thank all of those who have made this journey worthwhile.

First and foremost, thanks to God Almighty for his grace, wisdom and comfort throughout my thesis. There were times where things were very tough and God was there to protect and give strength to overcome heavy storms and strong waves. This was indeed a very long journey and I praise you God always.

I am particularly grateful to Dr Rodina Ahmad who is her excellent supervision, helpful guidance and continuous encouragement from the initial to the final stage enabled me to develop a deeper understanding of my research. I am grateful for having the opportunity to work under her supervision.

My deepest gratitude and love to my parents, Shahla and Abolfazl and my brother and sister, Navid and Nafiseh for their love, support, guidance and mostly believing in me throughout my study life and making sure that my dream becomes a reality. I would like to thank God for blessing me with such a wonderful family.

I would like to extend a special note of thanks to my friends and all people who helped me towards this achievement. I would like also to acknowledge all of the respondents who evaluated my research. I wish you all the best.
Table of Content

Abstract .......................................................................................................................................... iv
Acknowledgment ............................................................................................................................ v
Table of Content............................................................................................................................. vi
List of Figures .................................................................................................................................. x
List of Tables:................................................................................................................................... xi

CHAPTER 1: INTRODUCTION ................................................................................................ 1
1.1 Background ............................................................................................................................... 1
1.2 Problem Statement .................................................................................................................. 3
1.3 Research Objectives ................................................................................................................ 5
1.4 Research Question .................................................................................................................... 6
1.5 Scope of Research .................................................................................................................... 6
1.6 Research Approach ................................................................................................................... 7
1.7 Research Organization ............................................................................................................. 9

CHAPTER 2: LITERATURE REVIEW ................................................................................... 11
2.1 Introduction ............................................................................................................................. 11
2.2 Quality Assurance in Requirement Engineering ................................................................. 12
  2.2.1 The Importance of Quality Assurance ........................................................................ 12
  2.2.2 Requirements and Quality Assurance ......................................................................... 13
2.3 Software Inspection ............................................................................................................. 17
  2.3.1 Introduction to Software Inspection ............................................................................ 17
  2.3.2 Core inspection Concepts and Relationship ............................................................... 18
2.4 Learning Theories ............................................................................................................... 30
  2.4.1 Behaviourism ............................................................................................................... 30
  2.4.2 Cognitivism .................................................................................................................. 31
  2.4.3 Constructivism ............................................................................................................ 32
2.5 Blended Learning ................................................................................................................ 33
2.5 Teaching Framework .......................................................................................................... 37
2.6 Teaching tool ...................................................................................................................... 41
  2.6.2 The Classification of the Tools and Features of Collaborative Learning Application .. 44
  2.6.3 Shared Repository Tools ......................................................................................... 45
2.6 Conclusion .......................................................................................................................... 47

CHAPTER 3: RESEARCH METHODOLOGY ................................................................................. 48

3.1 Introduction ......................................................................................................................... 48

3.2 Literature Review ................................................................................................................. 49

3.3 Tool Development ................................................................................................................. 50

3.3.1 Inception .......................................................................................................................... 51

3.3.2 Elaboration Phase ............................................................................................................ 51

3.3.3 Construction Phase ......................................................................................................... 53

3.4 Evaluation ............................................................................................................................ 53

CHAPTER 4: DESIGN AND IMPLEMENTATION ...................................................................... 55

4.1 Introduction .......................................................................................................................... 55

4.2 System Overview ................................................................................................................. 55

4.3 Teaching Framework ......................................................................................................... 55

4.3.1 General Guidelines ......................................................................................................... 56

4.3.1.1 Software Inspection Concept and Principle ............................................................... 57

4.3.1.2 Software Inspection Technique .................................................................................. 58

4.3.2 Specific Guidelines ......................................................................................................... 58

4.3.2.1 Software Inspection Process and Roles ..................................................................... 59

4.3.2.2 Applying technique on Software Inspection Process ................................................. 59

4.3.3 Learning Activity ........................................................................................................... 60

4.3.3.1 Conceptual Samples: ................................................................................................. 60

4.3.3.2 Work Sample ............................................................................................................. 61

4.3.3.3 Collaboration in Practice ........................................................................................... 61

4.4 System Requirement ......................................................................................................... 62

4.4.1 Functional Requirements Specification ............................................................................ 62

4.4.2 Non-functional Requirements Specification ..................................................................... 64

4.4.3 Actors and Roles ............................................................................................................. 65

4.4.4 Use case ........................................................................................................................ 66

4.4.5 Use Case Description ..................................................................................................... 67

4.5 Class Diagram .................................................................................................................... 75

4.6 Sequence Diagram .......................................................................................................... 77
6.3.4. Evaluation of Results .................................................................................................... 106
  6.3.4.1 Ease of use .............................................................................................................. 106
  6.3.4.2 Teaching Framework Validation ............................................................................. 108
  6.3.4.3 System Functionality .............................................................................................. 110
6.4 Conclusion ............................................................................................................................ 112

CHAPTER 7: CONCLUSION.................................................................................................. 113
7.1 Goals and Objectives ............................................................................................................ 113
7.2 Research Contribution .......................................................................................................... 115
7.3 Future Research Works ......................................................................................................... 116
List of Figures

Figure 1.1: Research Methodology --------------------------------------------------------------- 8
Figure 2.2: Excerpt of quality assurance approaches (Aurum and Wohlin 2005) ------------------ 14
Figure 2.3: Dimension and sub dimension of identification classification ---------------------- 19
Figure 2.4: Blend of Learning Theories -------------------------------------------------------- 34
Figure 2.5: Ingredients of Blended Learning ---------------------------------------------------- 35
Figure 2.6: The practical teaching framework of ERP course ----------------------------------- 38
Figure 2.7: Conceptual sketch of framework illustrated as a concept map ------------------------- 40
Figure 2.8: The Sample of Moodle Interface ---------------------------------------------------- 42
Figure 2.9: The Sample of ATutor Interface ------------------------------------------------------ 43
Figure 3.1: Research Methodology --------------------------------------------------------------- 49
Figure 3.2: Rational Unified Processes --------------------------------------------------------- 51
Figure 4.1: Teaching Framework for software inspection ......................................................... 57
Figure 4.2: Use Case Diagram ---------------------------------------------------------------- 67
Figure 4.3: General Guidelines Sequence Diagram ----------------------------------------------- 76
Figure 5.1: Three-tier Architecture Model (Juel, et al., 2005) ---------------------------------- 78
Figure 5.2: Component Diagram For WBTSIS ----------------------------------------------------- 81
Figure 5.3: Package Diagram of WBTSIS -------------------------------------------------------- 83
Figure 5.4: Main Menu -------------------------------------------------------------------------- 85
Figure 5.5: Sign up ----------------------------------------------------------------------------- 88
Figure 5.6: Login ------------------------------------------------------------------------------- 89
Figure 5.7: Student’s Course ------------------------------------------------------------------ 91
Figure 5.8: Student’s Activity ----------------------------------------------------------------- 92
Figure 5.9: Student’s Selection ---------------------------------------------------------------- 93
Figure 6.1: Ease of Use Results ---------------------------------------------------------------- 108
Figure 6.2: Teaching Framework Validation ------------------------------------------------------ 109
Figure 6.3: WBTSIS tool Functionality ---------------------------------------------------------- 111
List of Tables:

Table 4.1: Functional Requirements .................................................................62
Table 4.2 Functional Requirements ........................................................................63
Table 4.3 Actors & Roles .........................................................................................65
Table 4.4: WBTSIS Use Case for Login Operation ....................................................67
Table 4.5: WBTSIS Use Case for Manage General Guidelines ....................................68
Table 4.6: WBTSIS Use Case for Create Software Inspection Concepts .......................69
Table 4.7: WBTSIS Use Case for Modify Software Inspection Concepts ......................70
Table 4.8: WBTSIS Use Case for Manage Conceptual Samples .................................71
Table 4.9: WBTSIS Use Case for Create Quizzes ......................................................72
Table 4.10: WBTSIS Use Case for Use General Guidelines ........................................73
Table 4.11: WBTSIS Use Case for Use specific Guidelines .........................................74
Table 6.1: Checklist on Administrator Screen Form Validation ....................................98
Table 6.2: Test Specification for Form Validation Function ......................................99
Table 6.3: Test case for Registration ........................................................................102
Table 6.4: Test case for Create Software Inspection Concepts ..................................102
Table 6.5: Test case for Update Software Inspection Concepts ..................................103
Table 6.6: Test case for Create Quiz .........................................................................103
Table 6.7: Test case for Create Project .....................................................................104
Table 6.8: Test case for View Group Practice ..........................................................104
Table 6.9: Summary of WBTSIS’s Ease of Use .........................................................107
Table 6.10: Summary of WBTSIS Teaching Framework Validation .............................109
Table 6.11: Summary of WBTSIS’s Functionality .....................................................111
CHAPTER 1
INTRODUCTION

1.1 Background

To deliver high-quality, performing software inspection is the industry’s best practice. Software inspection is an essential stage for managing software defects early in the development process (Sapsomboon, 1997). Software defects are some of the most important factors that affect the success of a software project. For over 20 years, software inspections have been described as one of most effective quality assurance techniques in software engineering (Laitenberger & DeBaud, 2000). The initial goal of an inspection is to detect defects early as defect heavily affects the improvement of the overall quality of software, with consequences that involve costs and time (Yourdon, 1997).

Software inspection, as originally defined by Michael Fagan over 20 years ago (Fagan, 1997), is recognised as an effective defect detection method. The basic technique is simple: a number of participants review a software document with the goal of discovering defects. The approach is effective because it involves people other than the author of the document. These participants are not familiar with the document; hence, they try to find more defects. Inspection can be used on any type of document, including specifications, designs, code and test plans. The inspection process originated by Fagan employs four to six people, each with specific roles. The moderator is the person in charge of the inspection procedures and carefully manages team members’ performance in each phase. In this case, the moderator is the key person in a successful inspection as he or she manages the inspection team and must offer leadership.
Special training for this role is suggested (Laitenberger and DeBaud, 2000). During the inspection meeting, a reader is required to correct the document and a recorder notes all faults found. The author of the document under inspection is another team member. Any remaining participants have the role of inspectors whose duty is to find defects in the software document. The selection of inspectors according to experience and background knowledge is considerable (Laitenberger and DeBaud, 2000).

The process used consists of five phases. During overview, the author describes the document that needs to inspection to the rest of the team. Each team member then does individual preparation which consists of reviewing the software document to obtain an understanding of it. Checklists are common techniques that can be used for finding defects to aid inspectors. Therefore, inspectors do not have clear instructions on how to use a checklist, that is, it is often ambiguous when to use it and it depends on what information an inspector can use to answer a particular checklist question (Laitenberger and DeBaud, 2000). An inspection meeting is then held. The reader makes some corrections in the software document, while inspectors raise any issues they have discovered. The team then discusses the issue until a consensus is reached. If an issue is agreed upon as a defect, it is classified and noted by the recorder. No attempt is made to find a solution to the defect; this is carried out later. After the meeting, the moderator writes a report detailing the inspection and all defects found, and the report is passed to the author. During rework, the author carries out modifications to correct defects detailed in the moderator's report. The moderator then performs a follow-up phase, ensuring that all required alterations have been made.
1.2 Problem Statement

Software engineering students are generally willing to help themselves to become critical thinkers and in order to acquire the abilities and skills about software engineering subjects such as software inspection to be familiar with its concepts and process.

The need for quality is increased when the popularity of software grows. In this case, software engineering researchers are motivated to investigate technologies that can help to improve quality. In many cases, software inspection is an essential part of the quality assurance effort for software products. Inspection is a process that it consists of checking by a group of people such as inspectors and moderator. The inspection process makes the assumption that each inspector has knowledge and experience to inspect documents. As there are various techniques and methods used within a software development process, it can cause inspector, who may be very experienced in a highly technical domain, to perform differently in a software inspection (Laitenberger and DeBaud, 2000). Therefore, organisations employ expert to perform the inspections in order to gain better outcomes from the inspection as lack of sufficient knowledge of software inspection among software engineers causes many problems to occur during the inspection process in organisations. Two important issues must be better understood.

Firstly, inspectors have difficulty understanding the concepts of software inspection as well as how to find defects, how to apply techniques and how to fix those problems. Secondly, it is important to understand other variables that affect the effectiveness of an inspection performance. Therefore, the organisation has to think about how the specific techniques and
methods will positively or negatively affect their software development process. Organisations also must use a way to evaluate the background knowledge and experience of each of their team members. In order to do this, background knowledge and the performance of their work during an inspection must be investigated. Organisations want to choose inspectors who are very familiar with the related domains to be members of their Software Requirements inspection team. Software engineering students are generally willing to help themselves to become critical thinkers and to acquire the abilities and skills that are related to software engineering subjects such as software inspection and its techniques. The goal of studying in a university is for student to prepare themselves for their future career. The problems mentioned are major issues in the software engineering in industry. The effective performance of the software inspection process requires some background knowledge as well as some experience in requirement engineering. Students also need to work in a group on software inspection projects and learn how to lead it, not only rely on context-based knowledge. As a result, the software engineers may be taken by surprise when they work in organisations as inspectors because they do not fulfill the organisation’s expectation and also cannot solve its problem. It is time consuming for organisation to ensure that software engineers are familiar with software inspection and its technique. Training them would raise the total cost of software development. A support tool is thus needed to teach software inspection which is designed to help software engineering students. Software engineers tend to understand the inspection process better by using concepts and definitions. The problem is that the average student does not have a sufficient body of knowledge and practical ability from which to source these software inspection instances. The many text books, websites and papers on software inspection include numerous examples but these examples are often trivial and the usage of patterns in simplified scenarios is seen as
insufficient. On the other hand, understanding the use of software inspection in the real world requires a deep knowledge of the context describing a situation where a particular inspection is useful. It is against this background that this researcher decided to design a support tool for teaching software inspection.

1.3 Research Objectives

This research attempts to solve problems that addressed in the problem statement. It plans to design a tool for teaching software inspection to software engineering students and investigates a possible way to educate them on software inspection. The research project includes the following objectives:

1. To develop the teaching framework for software inspection
2. To design and develop the appropriate support tool based on the proposed framework for teaching software inspection.
3. To evaluate the support tool for teaching software inspection.

In this research, the researcher initially studies the problems mention by existing papers about software inspection and its techniques. The next steps are to identify the problems of using software inspection and applying their techniques on projects carried out by software engineers. The research deals with finding an approach on how to teach software engineering students to apply software inspection and its techniques in their software projects and in order to achieve good knowledge experience and practical abilities.
1.4 Research Question

The following questions are aimed at finding out whether the software inspection teaching can help the students’ learning:

1) How can the tool improve the knowledge and practical ability of software engineering students regarding software inspection?

2) Can the teaching tool overcome the lack of knowledge about software inspection concepts and solve the difficulty of guiding teamwork for performing software inspection and its techniques?

3) What is the effectiveness and efficiency of this tool in addressing a student’s needs, skills and practical abilities?

1.5 Scope of Research

The main area of this study focuses on requirement engineering. In this research, the scope is limited to software inspection from the angle of teaching methods for software engineering students. The main concepts of this thesis are:

- The area of requirement engineering
- Teaching the inspection approach
- Framework teaching tool
- Focus on specifying the requirement document
The purpose of this thesis is to teach students about software inspection definitions and how to apply inspection techniques. The new teaching framework tool will be evaluated by the software engineering students for its usefulness in teaching practicalities.

1.6 Research Approach

This thesis plans to develop a software inspection teaching tool for the requirement engineering phase of software requirements. The process of this research is conducted through the following steps.

Firstly, the researcher reviews and summarised the philosophy behind software engineering techniques such as software inspection and related techniques in software development. Since software inspection is beneficial to the quality of whole software, the perceived advantages of software inspection is narrowed down to the software engineering students and also their lecturers.

Secondly, the researcher should studies several learning theories and reviews many teaching approaches. The researcher also investigates many teaching tools for increasing efficiency and effectiveness of the learning students.

Thirdly, based on the literature review and the summarized information about existing teaching tools, researcher designs a teaching framework. This framework is a guideline for software engineering students to improve their knowledge and practical abilities about software inspection while at still studying in the university and later on in their industry career.

Fourthly, based on collected data and functionality, the teaching tool is implemented. The teaching tool includes all interactive features and is suitable for web-based designs.
Finally, the tool is evaluated by students. Data from the testing are collected, tabulated, and documented to determine the tool’s strengths and weaknesses so that the system can be improved in the future.

![Research Methodology Diagram](image)

**Figure1.1: Research Methodology**
1.7 Research Organization

This thesis is structured as follows:

**Chapter 1: Introduction**

This chapter explains the introduction of this thesis and describes the background, problem statements and objectives.

**Chapter 2: Literature Review**

This chapter describes the major contexts related to the title of this thesis and includes an overview of the software inspection process as well as concepts and learning theories.

**Chapter 3: Research Approach**

In this chapter, the software process model of WBTSIS is described. The goal and scenario approach and function point extraction rules has been illustrated.

**Chapter 4: WBTSIS Analysis and Design**

In this chapter, the analysis and design of WBTSIS is described. The requirements analysis of the tool is conducted to identify its functional as well as its non-functional requirement.

**Chapter 5: WBTSIS Implementation**

This chapter presents the implementation of the WBTSIS tool and how it works. The implementation part of this chapter describes the tool’s elements, is main component and user interface.
Chapter 6: Testing and Evaluating

This chapter describes the testing and evaluation of tool. The first part of this chapter is to introduce the testing of WBTSIS. Then evaluation is described to determine whether the developed WBTSIS tool fulfill its requirements and its usability.

Chapter 7: Conclusion and Future work

This chapter summarizes and concludes the contents and contribution of this thesis.
CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

This chapter gives an introduction to the software inspection process and its techniques and to teach software inspection concepts to software engineering students using the Web-based tool which is considered a new and better environment for teaching. The definitions of a few terms are also provided. Quality assurance is explained in detail.

Software inspection has been carried out using by different types of dimensions, processes and techniques. Hence, some methods of teaching, especially collaborative methods, also are explained to enable better understanding of the interactions involved in teaching environments. This is done by discussing their theories, features, processes and techniques as well as by comparing these methods with the traditional way of teaching. In addition, this chapter also explains of a related teaching framework as a guideline to how to enhance learning of software inspection.
2.2. Quality Assurance in Requirement Engineering

2.2.1. The Importance of Quality Assurance

In today’s world, the demand for higher quality is fast increasing; the selection of appropriate techniques for delivering a software product on time and within budget is also difficult. Requirement engineering is a fundamental phase of the software development process, and all next phases of the development are affected by the requirements. This means that making quality requirements is a major factor in ensuring the overall quality of the developed system (Aurum & Wohlin, 2005).

Why should defects be identified early in the requirement engineering process? Defects create an issue because they can not only influence other requirements, they also introduce faults in the architecture, design, coding and testing. Quality assurance (QA) in the test and implementation phase depends on the abilities and experience of the requirement engineer, designer and programmer to make products of high quality. No intermediate QA in each phase means that designing and implementation are performed based on the wrong requirements and analysis. This will lead to high rework and revision of all the system architecture and design due to requirement defects (Aurum & Wohlin, 2005).

In addition, finding faults early in the development process is more cost effective than during testing or maintenance (Briand, Freimut et al. 2002). Resolving faults in the requirement phase involves less cost and effort rather than in other phases. A requirement issue can be more expensive if it is identified in operation, compared to detecting it in the requirement phase. Based
on these reasons, QA techniques for requirements are suitable techniques to ensure success and to avoid rework in later phases (Aurum and Wohlin, 2005).

2.2.2. Requirements and Quality Assurance

Depending on the organizational viewpoint, quality is a complex concept to describe (Saarinen & Vepsäläinen, 1993). For example, does high quality mean fewer faults per line of coding? What problems will these defects make in the life of the software? Quality is defined differently in various situations, especially in the requirements and its definition becomes more complex when it is based on various stakeholders' views. For example, if the system is not completed based on the stakeholders’ needs, the system is seen as not having high efficiency and it might not support users’ needs and their particular tasks. Quality and quality assurance can be specified for requirements and they demonstrate aspects of defining a quality strategy for early QA.

It is essential to introduce the quality strategy to verify and easily validate quality issues strategy in the requirements phase. The objective of this strategy is that it helps to obtain the desired quality at the desired cost. The possible quality assurance approaches and methods can be used to check various quality attributes of the requirement in the quality assurance strategy. Aurum and Wohlin (2005) said the QA approaches are the technical major of the quality strategy to demonstrate the means of obtaining valuable requirement quality. The following framework expresses the meaning of a valuable quality assurance strategy. It demonstrates that elements are
significant for quality assurance in the requirement engineering phase. Appropriate quality assurance techniques that act best in specific situations using this framework should be identified. QA techniques (QA approaches) are the most fundamental element in the framework. It should be applied before explaining the QA strategy’s details as it is essential to analyse the main approaches for investigating the quality attributes of the requirements.

Figure 2.2: Excerpt of quality assurance approaches (Aurum and Wohlin 2005)
2.1.2.1 Constructive Approaches

While making the requirements, quality is expressed through constructive approaches. This means that the goal of constructive approaches is to minimise mistakes. These approaches are applied during the development of the requirements. Constructive approaches include:

1. **Elicitation Techniques**: The elicitation stage is the initial phase to check quality generally and select the final system (Leffingwell & Widrig, 2000). In this process, various stakeholders communicate with each other to agree on appropriate requirements. To discover the requirements, engineers should apply different techniques to support stakeholders’ interests.

2. **Specification Techniques**: The goal of this stage is to collect the requirements in an appropriate way such that they can take advantage of the development. The result of this particular activity is usually a requirement document that includes the related factors of the system to be implemented, such as functional, non-functional aspects, assumption and restrictions.

3. **Prototyping**: the use of prototypes can be considered as an approach to support elicitation. The aim of a prototype is to enable the stakeholders to interact with the system and to give their ideas about how the system should work (Sommerville & Sawyer, 1997). By doing this, they understand the system’s needs better and thus it helps to unearth any missing requirements. The main point of a prototype is to complete the gap between explanation and implementation (Endres & Rombach, 2003).
2.1.2.2 Analytical Approaches

The analytical approach evaluates the requirement specification, whether the indicated requirements in it meet the quality criteria specified. The main output of this approach is that there is no reliable documented source. The stakeholders of the requirements should be concerned with QA. The next step, the requirements inspections and test case creation (as a part of acceptance testing), are explained in more detail below (Aurum & Wohlin, 2005):

- Requirement Inspections: There are many empirical and industrial outputs that describe inspection as having valuable meaning correctly expressing the quality of software correctly (Thelin, Runeson et al. 2003). The highest expense of the requirement issue is related to requirements inspections. Thus detecting defects at this stage is one of the most cost effective QA approaches. It prevents issues from being moved from the requirement phase to other phases and minimises rework (Briand, Freimut et al. 2002). Another important benefit of early QA is help to perform early QA activities such as inspections and test case creation. An inspection is identified by a process, reading techniques used, and the information on how the outcomes of the inspection are documented (Laitenberger 2000).

- Requirements-Based Testing: When executable parts of a system are ready, testing can be applied at the end of the development process. Testing will validate whether the system meets its specification. Testing can also perform fast searches for failures in the final software. The test planning and test case creation should at least be combined to use in the development process (Aurum & Wohlin, 2005).
2.3. Software Inspection

2.3.1. Introduction to Software Inspection

The main concept of inspections is to find faults in the software document. Inspection helps to ensure that the artifact has sufficient quality. Since the first finding on inspections in 1976 (Fagan, 1976), many variations have emerged. The main objective of inspection was defined by Fagan. For over 20 years, software inspection has been proposed as a simple, cost-effective technique for defect detection in all types of documents such as requirement, design, code and test. The primary aim of an inspection is to find faults early and then enhance the general quality of software within the desired time and budget to achieve best results (Laitenberger and DeBaud 2000). Inspection introduces an approach that includes a well-defined process and it can analyze a software product using a specific technique such as reading technique to satisfy the objective of identifying faults. It means that the author can apply inspection on a requirements document, design, or code.

An inspection consists of different steps. The inspection leader first chooses a team of people who will execute the inspection process. Then, the document to be inspected is divided between team members. The author of the document gives a general explanation about the system so that team members understand the general background. The team members then individually spend time reading over the document and becoming familiar with it. When all inspectors are ready, the inspection leader prepares a meeting. At this meeting, the document is read and faults are revealed by the inspectors. The faults are documented, and the document sent
back to the author. The author then revises these faults, or explains why they are not faults (Laitenberger and DeBaud, 2000).

2.3.2. Core inspection Concepts and Relationship

Classification is performed to express the core concepts and relationships of software inspection. This is based on five basic dimensions, namely: technical, managerial, organizational, assessment and tools. These particular sub-dimensions are described as important for software inspection. Each dimension and its relevant objectives are briefly described below.

The main purpose of the technical dimension is to describe in more detail the software inspection process, the different team roles, the essential size and selecting the technique to identify faults in the software product (reading technique). The managerial dimension demonstrates the effects of the information gathered during the inspection on the project. Managers are eager to use the inspection approach that affects the project effort (effort) and project duration (duration). The organisational dimension shows the effects of inspection on whole organisation and vice versa. This dimension analyses project members (team), and the environment (environment). Sub-dimensions are considered essential information in the inspection context. The assessment dimension consists of qualitative and quantitative evaluations of inspections. Finally, the tool dimension shows how tools can help in inspections (Laitenberger & DeBaud, 2000). Figure 2.2 shows the dimensions and sub-dimensions of the identification classification.
2.3.2.1. The Technical Dimension of Software Inspection

This dimension of an inspection’s classification consists of sub-dimensions in the inspection process, the team roles and size as well as the reading methods in an inspection. The sub-dimensions are described in more detail in this section (Laitenberger & DeBaud, 2000).
i. The Process Dimension

Software inspection contains six important steps: planning, overview, defect detection, defect collection, defect correction, and follow up. However, these steps are common for the inspection process (Laitenberger & DeBaud, 2000).

a. Planning

This phase consists of defining objectives, identifying inspection participants, their assigning roles and making a schedule for inspection meetings. Planning is considered as a distinct phase because a person should be responsible for planning in all levels of the inspection process (Laitenberger & DeBaud, 2000).

b. Overview

The fundamental objective of overview is to make inspections easier and to make it clear for participants to understand. The first meeting is valuable to inspection because this is when the author gives the assumptions and background to other participants. This phase may consume effort and increase the duration of an inspection. Other authors believe that this phase is the initial phase for executing the next inspection phases. There are two considerations that make the meeting more useful. Firstly, the document to be inspected may contain some difficult aspect which needs to be understood. Therefore, the author’s explanation about the inspected document helps the inspection participants to understand it and they become familiar with the inspection. Secondly, the inspected document is part of a large software system. In this situation, the author’s explanation may guide other participants to execute their duties more effectively and save time in the inspection phase (Laitenberger & DeBaud, 2000).
c. Defect Detection

This phase can be mentioned as one of the most important concept of an inspection process. The main objective of this phase is to examine a software product in order extract defects. Each inspector analyses the document to become familiar with it and individually find possible defects. Fagan (1976) states that a group meeting gives positive results, that is, most defects are found because inspection group members discuss and analyse the document together.

d. Defect Collection

Many team members participate in the inspection process to examine a software document in order to identify defects. Hence, each inspection participant should collect and record the defects detected. Then, a decision must be made on whether a defect is really a defect. This is done at meeting during which the inspectors discuss the defects from the individual reviews and examine the document to find further defects (Laitenberger & DeBaud, 2000). These are the main goal of the defect collection phase.

e. Defect correction

In this step, the author revises and fixes defects detected by the inspectors (Fagan, 1976). He or she makes corrections to the inspected document and investigates each reported defect.

f. Follow-Up

The main concept of this phase is that the moderator investigates whether the author has revised all found faults. The moderator then gives final suggestions for the collecting process and the product data for the proposed quality improvement. Doolan (1992) explains that the moderator verifies that the author has made some correction for each defect found.
ii. The Team Role and Size

Practitioners are usually interested in answering the following three questions about software inspections: (1) what roles are involved in an inspection, (2) how many people are selected for each role, and (3) how to select people for each role. To answer the first question, each inspection participant has a clear and specific responsibility. The roles and their responsibilities are explained in more detail by Ackerman (1989), Fagan (1976) and Russell (1991).

- **Organiser**: she/he manages all inspection process throughout the duration of the project.

- **Moderator**: The moderator handles inspection steps and meticulously oversees how team members do their duties in each phase carefully. He or she manages the inspection meeting. In this case, the moderator is mentioned as the key person because she/he has a significant role in the success of the inspection process. Therefore, particular training for this role is needed.

- **Inspector**: the inspector’s responsibility is to identify faults in the software document. All inspection participants can be usually described as inspectors regardless of their specific roles.

- **Reader/Presenter**: During the inspection meeting, the reader will guide the team and explain and interpret the document.

- **Author**: She/he should improve the inspected document and should fix faults during correction. During an inspection meeting, he or she relies to specific questions that the reader is not able to answer.

- **Recorder**: During the inspection meeting, the recorder should collate faults in an inspection list.
• **Collector**: The collector gathers all faults detected by the inspectors if there is no inspection meeting.

The second question to be considered is how many people are to be selected for each role. Assigning resources to these roles depends on experience. Fagan (1976) suggests keeping the inspection group small, to almost four people. Bisant and Lyle (1989) found performance advantages in an experiment involving two persons: one inspector and the moderator, who can also be regarded as an inspector. Madachy, al. (1993) present data showing that the optimal size is between three and five people. The answer to this question is not clear and it does not depend on the type of product or the environment where the inspection is performed. Some people believe that it is better to start an inspection with three to four members, including author, inspector and moderator.

The last question to consider is how to choose the participants of an inspection team. Candidates selected for the role of inspectors are personnel involved in product development (Fagan, 1986). During a project, outside inspectors can be added as participants because of their particular knowledge and previous experiences. Selecting of inspectors based on experience and background knowledge has recognise two major considerations. The first is that inspection outcomes are based on human factors. This often limits the lack of related inspectors. Secondly, participants with less experience and knowledge are not selected as inspectors.
iii. The Reading Technique Dimension

The most vital and difficult part of requirement inspection is finding defects. In this part, the inspectors find faults. To perform this step, the inspector can apply a reading technique for supporting inspection (Aurum & Wohlin, 2005). A reading technique demonstrates a sequence of stages that guides an inspector in gaining a deep knowledge of the requirements related to inspection and identifying issues related to them (Laitenberger, 2001).

The inspector can meticulously identify defects carefully by applying particular techniques in software documents that may enhance the efficiency of an inspection team. One of most important techniques is a reading technique (Laitenberger & DeBaud, 2000). Which is a method that each inspector to identify faults in the inspected document. The ad-hoc reading, checklist-based reading and scenario-based reading are seen as the most popular reading techniques used today for finding defects in inspections (Fagan, 1976; Gilb & Graham, 1993).

When a software product is submitted to inspection participants that do not contain any outlines on how to perform an inspection or what information to seek, the technique of ad-hoc reading helps to support the inspector’s examination of the document. The word 'ad-hoc' only means that it does not give inspectors any guidelines for solving the problem of how to identify faults in a software document. In this situation, identifying faults is completely based on the skill, background knowledge and experience of an inspector.
Checklists contain questions and support inspectors in that they try to answer specific questions during their reading of the document. These questions concentrate on specific quality approaches related to the requirements under inspection. Firstly, the checklist approach guides an inspector on what she/he should verify (Aurum & Wohlin, 2005) and helps the inspector to understand more about the inspected document. It can be fundamental to find exact faults. Secondly, deciding on when to use a checklist to best advantage is usually difficult and complicated and depends on what information an inspector can used to answer a specific checklist question. Checklist-based reading has many shortcomings such as the fact that the questions of a checklist are often limited to the finding of faults which are part of specific defect types.

Another reading technique used is the Scenario-based reading. The core concept of scenario-based reading is the benefit that viewing scenarios will guide inspectors to identify faults. A scenario may include a sequence of questions or a more detailed explanation to aid an inspector in performing document review. A scenario focuses the concentration of an inspector on finding defects as guideline. Used by an inspection team, scenario-based reading is more effective but the rate of its effectiveness depends on the concept and design of the scenarios (Laitenberger & DeBaud, 2000).

2.3.2.2. The Managerial Dimension of Software Inspection

In selecting a particular inspection, effort is expended. This dimension is seen as the managerial dimension. This phase assesses whether it is worthwhile spending effort on
inspection and also mentions how inspections influence on software product as well as the cost and the duration of the project.

I. Cost

Since inspection is a human activity, a project manager should have a proper understanding of the cost involved in inspections. Most research researches has have found that the costs for finding and deleting defects during inspections is much lower than finding and deleting the same defects in later phases. The cost depends on the size of the inspected product or the number of faults found. (Ackerman, et al., 2002)

II. Duration

Inspections do not only involve expending effort, but they also have an effect on the time taken. Inspection phases are based on a timetable which all the people involved in the project have committed to that they can attend and perform their roles. Sometimes, duration is a critical factor if time to sell of product is a critical issue during development (Laitenberger & DeBaud 2000).

2.3.2.3 The Organizational Dimension of Software Inspection

The inspection process is introduced as new process in organisations. Hence, it affects the whole Organisation, which includes work teams and the environment.
i. Team

Human activity is the main factor of software inspection. Software inspection is analysed by the inspection participants. Hence, the status of whether the software inspection is a success or a failure depends on quality enhancement and whether costs increase heavily depends on human factors. Training allows project members to have their own ideas on how inspections are performed and how main defects can affect the environment for further process improvements (Laitenberger & DeBaud, 2000).

ii. Environment

The inspection process is defined as technology for transferring initiative. The issue of organization, it is also not just in terms of its participant but also in terms of its culture, management, and budget, quality, and productivity objectives. All these factors can be included in the sub-dimension of organisational environment.

2.2.2.4. The Assessment Dimension of Software Inspection

The based used for evaluation an inspection on whether it gives any benefit is different between qualitative assessment and quantitative assessment. While qualitative assessment is often based on the personal ideas of inspection members, quantitative assessment is based on the information collected in inspections and the subsequent step of identifying defects. In comparison with the managerial dimension, the assessment dimension describes how inspections are evaluate inspections rather than the outcome of the evaluation (Laitenberger and DeBaud 2000).
i. Qualitative Assessment

Qualitative assessment is based on inspection profit rather than on data. Weller states that inspection members must have proper understanding of the software process and the developed product (Weller, 1993). Furthermore, inspectors try to guide participants as they perform group work because they prepare the teams to observe each other's strengths and weaknesses (Doolan, 1992; MacLeod, 1993; Franz & Shihs, 1995) One observation is that inspection participants have a clear understanding of the whole system, not just of the role which each of them plays in the inspection process. (Laitenberger & DeBaud, 2000).

ii. Quantitative assessment

Quantitative assessment is an evaluation model that can be applied on the inspection data. It presents how to merge the inspection data to have a valid result. Most of the models do not mention about the cost of executing inspection (Fagan, 1997; Jones, 2002; Remus, 1984). These models are fundamentally related to the number of defects detected in inspection and to the total number of defects in a software product (if available).

2.2.2.5 The Tool Dimension of Software Inspection

Currently, a few supporting tools for inspection are available. Developed tools investigate software (often source code) inspection by researcher. Some inspection tools are classified in the following discussions: (1) ICILE (Brothers, Sembugamoorthy et al.,1990) is a unique tool for
making use of knowledge to help in finding logic defects; (2) Scrutiny (Gintell, Houde et al. 2002) is a tool based on the inspection method used at Bull HN Information Systems; and (3) CSI (Mashayekhi et al. 1993) supports the holding of synchronous, distributed meetings for development team which are geographically separated during the inspection process. In general, all tools prepare more or less suitable document handling facilities for browsing documents online. However, the use of these tools has limitations related to the particular burden of specific development situations.

i. ICILE

ICICLE can support the inspection of C and C++. This tool supports both individual preparation and the inspection meeting. During the inspection meeting, the tool provides the functionality required for individual checking, supplemented by support for collaboration. (Brothers, et al., 1990)

ii. Scrutiny

This process consists of four parts. The first stage is initiation and is the same as with the overview of Fagan’s model. The second stage is preparation, as in the Fagan model. The inspection meeting itself is called resolution, while the final stage, completion, includes both rework and follow-up. The roles of each inspection participant are also similar; however Scrutiny also implements some differences. Firstly, the moderator’s role is different and consists of a reader’s duties. In addition, the recorder role can be taken on by more than one person. Scrutiny also clearly implements the role of the producer who can answer questions based on the document. Finally, there is another role in the form of the moderator who ensures that the defects found by the inspection participants have been correctly addressed by the author. This role may
be assigned to any participant. Any other members of the team can take the role of inspector. Each stage of the process, along with each of the three roles, is modeled in Scrutiny. (Gintell, Houde et al. 2002)

iii. CSI

This tool was developed by Vahid Mashayekhi at the University of Minnesota. He created three prototype inspection support tools. The first of these, Collaborative Software Inspection (CSI) (Mashayekhi, al., 1993), is to support inspection of all software products. Using tool, each inspector can individually make a list of defects during inspection, which are then given to the author of the document before the inspection meeting. It is the author’s task to relate these defect lists and to then address each defect at the inspection meeting.

2.4 Learning Theories

This step discusses the basic learning theories of Behaviourism, Cognitivism and Constructivism and their affect on e-Learning. The goal of learning theory is to design an educational system that transfers content, knowledge and skills in a clear, well-structured and understandable manner. It is this view that provides most curriculum and instruction.

2.4.1 Behaviourism

This theory is comparatively easy to understand because it is based on observable behavior and can explain several general laws of behavior. Behaviourism is often used by teachers, who motivate or punish student behaviours. Its advantages are seen when used to support adult learning to get the desired behavioural outcome. This theory believes that the teacher’s job is to
make situations which create desired behaviour to their students. The behaviorist expects the teacher to know all the skills which they believe are needed for the students to learn and then present them to the group in a sequenced manner (Skinner 1986).

2.4.2 Cognitivism

This theory is defined as symbolic constructions in the minds of learners, and learning becomes the process of committing these symbolic representations to memory where they may be processed. This theory highlights the importance of the sense of learning about new information received; the teacher creates a central role for sending the correct information which the learner should understand and learn. The function of learning in cognitivism is to make an exact image of the external world in the learner’s minds. The objective is still to learn in the most effective behavior, but to break the lesson down into different parts that move from the simple to the complex in order to build on the learner’s previous schema.(Ertmer and Newby 1993)

i. Gagne’s Learning Theories

This is defined as a hierarchy of intellectual skills that are classified based on complexity so as to identify requirements for improving learning step by step. Instruction can be more effective by conveyed through a sequence of nine instructional events defined as intellectual skills that the learner is required to learn for the specific task. Gagne outlines the following nine instructional events (Patsula, 1999):

1. Gaining attention (reception)
2. Informing learners of the objective (expectancy)
3. Stimulating recall of prior learning (retrieval)
4. Presenting the stimulus (selective perception)
5. Providing learning guidance (semantic encoding)
6. Eliciting performance (responding)
7. Providing feedback (reinforcement)
8. Assessing performance (retrieval)
9. Enhancing retention and transfer (generalization)

2.4.3. Constructivism

This theory includes the attention, encoding and retrieval of knowledge processes from cognitivism, but maintains that there is no single exact representation of the world, only interpretations of experience. As far as teaching is concerned, the teacher or instructor should try and to motivate students to find principles by themselves. The role of the instructor is to translate and transform information to be learned into an appropriate format for the learner to better understanding. A number of principles are required to achieve efficient instruction and learning within the scope of the constructivist theories. The following theories are located in this domain:

i. Vygotsky’s Theory

This theory posits that social interaction plays an initial role in the development of cognition. Instruction can be made more efficient when learners engage in activities within a supportive environment and receive guidance mediated by appropriate tools.
ii. Bruner’s Constructivist Theory

Learning is an active process in which learners create new ideas based on their background knowledge. Instruction can be made more efficient by providing a careful sequencing of issues to allow learners to understanding and gained information to discover the key principles by themselves.

2.5 Blended Learning

The blended learning is defined as a combination of different learning environment. It gives learners and teachers a potential environment in which to teach and learn more effectively. Blended learning increases the options for greater quality and quantity of human interactions in a learning environment and blended learning offers learners the opportunity to learn together. With today's prevalence of high technology in many countries, blended learning often refers specifically to the provision or use of resources which combine e-learning (electronic learning) or m-learning (mobile learning) with other educational resources, also called hybrid courses. Blended learning is learning that is facilitated by the effective combination of different modes of delivery, models of teaching and styles of learning, and is based on transparent communication amongst all parties involved with a course. Blended learning can be drawn from many instructional theories and different theories apply to different situations. Generally, people chose BL for three reasons:

- Improved pedagogy
- Increased access and flexibility
- Increased cost-effectiveness
The instructional design model is well defined for the concept of blended learning. By applying learning theories of Keller, Gagné (1997), Bloom, Merrill, Clark and Gery (see Figure 2.3), five key ingredients emerge as important elements of a blended learning process (see Figure 2.4):

1. **Live Events:** Synchronous learning happens when participants learn at the same time in learning events, such as in a live “virtual classroom”.

2. **Self-Paced Learning:** Learners individually perform and complete duties, at their own speeds and on their own time, such as in interactive, Internet-based learning.
3. **Collaboration:** This happens in environments in which learners collaborate with each other, for example, e-mail, threaded discussions or online chat.

4. **Assessment:** The learners’ knowledge can come from self-paced events in which learners decide about previous knowledge, and post-assessments can occur during the live or self-paced learning events.

5. **Performance Support Materials:** These are on-the-job reference materials that enhance learning maintenance and transfer, including PDA download, and printable references.

![Figure 2. 5 Ingredients of Blended Learning](image)

**Ingredient 1: Live Events**

Live, synchronous events are a major ingredient of blended learning. This ingredient effectively highlights a learner’s understanding, such as keeping his/her attention, confidence and satisfaction.
**Ingredient 2: Self Learning**

Self learning adds significant value to the blended learning. Actual implementation of instructional design is widely different and has various outcomes. For example, two products may both be “based on” Gagné’s hierarchy that includes stated objectives, scrolling text and also technical animations, MP3 quality audio, and search ability to the mix. In implementation, multimedia design theory can be used.

- **Multimedia and Modern Design Theory**

  The use of multimedia as a tool to improve knowledge transfer is defined in instructional design.

  1) The multimedia principle: adding graphics to text can improve learning

  2) The contiguity principle: placing text near graphics improves learning

  3) The modality principle: explaining graphics with audio improves learning

**Ingredient 3: Collaboration**

The primary objective of collaborative learning is the completion of a group activity or project that differs in subject matter and discipline. Aside from science related subjects, other disciplines can also be collaborative in nature. As pointed out by Nobel et al. (2002), the sites of collaboration can be online (for example students chatting to peers on the assigned topic), in a laboratory (for example. students doing experiments in biology or chemistry), in the field (for example students collecting data for a short research on calorie consumption of McDonald’s customers, and in the classroom (for example students finalising their observations of different types of plant cells). It may also involve cross-curricular projects among different students from neighbouring secondary schools for the purpose of increasing the samples of their data in order
to be able to formulate conclusive statements about their project. During blended learning creation, designers should create learning environments in which learners and instructors can collaborate synchronously in chat rooms or asynchronously using e-mail.

**Ingredient 4: Assessment**

The most critical ingredients of blended learning is assessment. It is important for two reasons: 1) it ensures that learners are tested on subject about which they already know, gaining their own blended learning experience; and 2) it measures the effectiveness of all other learning ways and events.

**Ingredient 5: Performance Support Materials**

Performance support materials form an important ingredient of blending learning. In Gagné’s terms, they improve “learning retention and transfer” in respect of to the work environment (Gagne, 1977).

### 2.5 Teaching Framework

This part begins by describing the existing teaching framework to support software inspection. There are three main types of teaching frameworks that help student’s learning.

#### 2.5.1. Teaching framework of ERP courses

To help students gain a deep understanding of ERP and to improve their practical abilities, this framework is designed as a teaching framework of ERP courses. The framework consists of three parts: experiments in the lab, extracurricular activities and internship in a certain software company. Furthermore, there are three levels of experiments in the lab: business scenario simulation, business process and roles collaboration designed according to Zachman’s enterprise
architecture framework. After the experiments in the lab, students will get a training of ERP process and can construct a basic understanding of it. Extracurricular activities include the research of ERP business game, the simulation of development and testing of some ERP modules, and the implementation solution design of ERP. These activities will help students to improve ERP skill such as understanding about enterprise management and the business process. Software development and testing, etc. Students’ jobs during internship may be system development, system testing, system implementation and customer service. They will obtain guidance from experienced engineers in the organisation. The framework has been applied in real teaching environments and has gotten good results (Deng, Yin et al., 2009).

![Figure 2.6 The practical teaching framework of ERP course](image)

### 2.5.2. Skill based framework for industry career

A common complaint by industry professionals is the lack of knowledge of Software Engineering (SE) graduates. The lack of skills of many new graduates directly affect on
industry’s progress. For example, some of the graduates do not possess communication and managerial skills. Internship in huge industry based heaven projects are motivation for undergraduates who are in the penultimate and final years of their degree courses as they prepare to embark on their professional careers. Although, the goal is to provide students with a real-life SE experience and to prepare them for industry, these projects failed in some cases to provide the necessary breadth of skills. This failure motivated the researcher to develop a teaching framework based on skills. It focuses on managerial, engineering and personal skills. This new experience shows that this framework guide student learning during projects. Student portfolios and assessment artifacts clearly showed that a majority of the students highly benefitted from being taught using the framework. High scores in student feedback questionnaires indicate that the skill-based framework helped to guide student learning outcomes through specifying clear objectives. Overall, the application of the framework was a success from the perspective of students, staff and clients and fulfilled its major purpose of providing students with an adequate breadth and depth of software engineering skills. In the coming years, the researcher intends plan to improve the framework and its implementation to address some teething problems experienced and reported in this paper.(Karunasekera and Bedse, 2007)

2.5.3. A Teaching Framework for Essential Topics in Electromagnetics

Teaching undergraduate engineering electromagnetics (EM) requires extensive presentations of the basic theoretical and fundamental physical concepts that support most electrical engineering principles (Cheng et al, 2003). The following challenging components of teaching undergraduate EM courses were identified.
Firstly, the technical level of the material makes it difficult to teach students on how to use basic electromagnetic principles. Secondly, motivating students to learning EM poses many problem. Finally, in the process of learning the introductory essentials, it is not easy to link the concepts they are learning with applications and practice. Therefore, a teaching framework proposes to help overcome the mentioned obstacles and challenges encountered in instruction of introductory EM courses.

The framework creates connections between theoretical fundamentals and pedagogical examples, and between historical perspectives and courses and the practical applications found in industry (Giannacopoulos and Popovic, 2005).

Figure 2.7 Conceptual sketch of framework illustrated as a concept map
2.6. **Teaching tool**

Some teaching tools can enhance the learning process. They are defined as communication and collaboration tools that can be used to engage students in learning together with others within a collaborative environment. Effective lecturing is a difficult task for many instructors attempting to deliver complex and voluminous amounts of information to students. Instructors use various techniques to make the lecture more interesting and yet communicate the important points of the subject matter. Some of these techniques involve the use of a classroom blackboard, a projector for static and animated displays, multimedia and collaboration, among others. Collaborative software, which is referred to as groupware or workgroup support systems, is a sort of software designed to help people who are involved in a common task to achieve their goals. It is the basis for computer-supported cooperative work. In educational collaboration activities, there are many collaboration types of software used, such as ATutor, Blackboard Inc, eCollege, Dokeos, eFront, Fle3, GCompris, Granule (software), ILIAS, LON-CAPA, Moodle, OLAT, SharePoint, Joomla, Renaissance Place and Sakai Project. The reasons that Atutor, Moodle and Sakai systems are chosen include their wide usage in educational arenas such as universities, high schools and institutes. In addition, these systems are free and open source. They also consist of web-based and virtual learning environments.

In this part of the research, the three systems are briefly introduced and then in Chapter 5, the new teaching tools and features are demonstrated. This research focuses on what is important from the learners’ point of view and what these systems lack. These systems are described below.
2.6.1 Collaborative Learning Applications Software

2.6.1.1 Moodle

One of a free and open source e-learning software platform, also known as a Course Management System, Learning management System (CMS), or Virtual Learning Environment is Moodle. It is designed using sound pedagogical principles, to help educators make effective online learning communities with opportunities for rich interaction. Moodle is open source and modular design means that everyone can develop additional functionality. Development is undertaken by a globally diffused network of commercial and non-commercial users, streamlined by the Moodle Company based in Perth, Western Australia (Zhang, 2009).

![The Sample of Moodle Interface](image)

Figure 2.8 The Sample of Moodle Interface
2.6.1.2 ATutor

One of an Open Source Web-based Learning Content management System is ATutor. It is used in various fields, involving online course management, continuing professional development for teachers, career development, and academic research. ATutor is referred to as unique for its accessibility features, and for its suitability for educational use according to software evaluation criteria established by The American Society for Training and Development. The software is used internationally and has been translated into many languages. ATutor is distributed under one of the OSI-approved licenses.

![The Sample of ATutor Interface](image)

Figure 1. The Sample of ATutor Interface
2.6.2 The Classification of the Tools and Features of Collaborative Learning Application

Synchronous communication is a tool that enables the users or correspondents to exchange information at the same time or in real-time. Since both users are online, this communication exchange is likened to a conversation. The main features of asynchronous communication are chat, shared whiteboard, application sharing, virtual space, voice chat, and video conferencing.

Chat is a specific feature in which users communicate using real-time text exchanges. The text exchanges may be archived and searched. In some instances, a moderator may either choose the topic or supervise the discussion.

Shared whiteboard refers to the shared text window which may include graphics, pictures, and drawings.

Application sharing is a tool that enables the user to run an application in a computer while the window he or she is running is simultaneously applied across the Web. It is also possible to share mouse control in application sharing.

Virtual space is like a physical place consisting of virtual meeting rooms of more than one discussion group using real-time text exchanges. In this space, the time and topic are usually but not always, predetermined. A moderator is optional, and sometimes guests may share the space.
Voice chat also called audio conferencing and pertains to an audio communication tool that uses real-time audio exchanges involving two or more persons.

Video conferencing is a facility that broadcasts video with real-time Q&A (question and answer) for immediate feedback and instructor floor control.

2.6.3 Shared Repository Tools

Shared repository tools are categorized into two groups: learners and teachers.

2.6.3.1 Shared Repository Tools for Learners

Shared repository tools for learners facilitate sharing of resources as well as provide archival storage of learning products (Koschman, 1996). Through these tools, learners have the control to select tasks and become responsible in sharing and learning (Tinzman et al., 1990). The tools for shared repository have six main features. These include the following:

Shared Facility searches documents using particular keywords and subsequently returns a list of documents indicating the links where the keywords can be found.

Curriculum Objectives/Syllabus are composed of course outline, course structure, timetable, and other important dates for the students to use in order for them to be guided as they plan their activities related to the objectives of the course.

Shared Bookmarks enable the users to create, display, manage and update specific sites. By sharing these features, users create a knowledge base for their personal needs as well as contribute to their group. Each user can share his or her bookmark. In bookmarking, suggested resource archives or links to hot lists may be available.
Real-time data come from the primary source. Data from credible sources are generally online and up-to-date.

Data collection pertains to the facility of creating a customised window to take the input of a user. A simple online voting poll is a good example.

Lists of Projects/Courses/Activities are typically arranged according to subject areas, age, curriculum level, and duration, and starting date. This format makes it possible for a search to be made using a Search engine. Many web sites provide the lists prepared by organisations that are doing collaborative activities.

Frequently Asked Questions (FAQ) is specifically a repository of questions that users ask. Answers to these questions are already predetermined.

2.6.3.2 Shared Repository Tools for Teachers

Collaborative activities for students are planned and implemented by the teachers. The collaboration is, therefore, is initiated among teachers, moderated and finalised among themselves before sharing them with the students (Koshmann, 1996). In basic, the shared repository tools for teachers include Design Activities.

Design Guidelines for collaborative learning include lesson plan, and task formats or different learning activity formats. The layout of the design process can be done in the form of charts and templates made available for teachers to create lessons and other activities.
Shared repository tools for learners improve sharing of resources as well as provide archival storage of learning products (Koschmann 1996). Through these tools, learners have to concentrate on selecting tasks and become responsible in sharing and learning (Tinzman et al., 1990).

2.6 Conclusion

In this chapter, some existing researches pertaining to the software inspection process, related techniques, teaching tools and frameworks are reviewed, and major findings of teaching software inspection tools are mentioned. The purpose of this chapter is to find existing knowledge associated with software inspection. Reviewing current literature indicated that, previous researchers did a lot of research in this field but there are still a lot of difficulties and complexities. However, the main goal of teaching software inspection is for the participants to learning the concepts. During the inspection process, Laitenberger and DeBaud (2000) suggest that special inspection team needs special training because experience and knowledge are the two main factors to achieving the goal of inspection. Furthermore, defect detection fully based on the skill, the knowledge, and the experience of an inspector.
CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This part describes a methodology for the development and documentation of this dissertation, and to define the techniques and software process used for development and implementation of a teaching tool to support software inspection. This tool is named as WBTSIS, an acronym used to denote the first letter of each word of the Web Based Teaching Software Inspection System. This part presents the techniques and software process model used for development and implementation of the teaching tool. It also provides a step-by-step development methodology, which is Rational Unified Process (RUP). Figure 1 illustrates the overview of the entire research methodology process.
3.2 Literature Review

In this step, the researcher focuses on a wide review of existing related literature to propose methods which can be useful in this research. This literature review is carried out in two main areas which are requirement engineering and software development. In particular, the review is focuses on software inspection and teaching framework, teaching tool and learning theories. In the first step, the researcher focuses on existing related literature about software quality, software inspection and its techniques in finding defects. The next step evaluates the existing teaching framework, teaching tool and current learning theories, and their features and characteristics are investigated in order to design tool for teaching software inspection. After studying all available
related resources to this research, analysing the concepts and gathering all relevant information required for conducting this research, the researcher finalises the literature review report and reviews the data and information needed for developing the teaching framework and teaching tool.

3.3 Tool Development

Tool development is the product of this research to help software engineering students to learn software inspection concepts and its techniques to readiness for their future careers. The WBTSIS tool is carried out, starting with general and specific guidelines such as software inspection process and the different types of software inspection techniques. Students can also perform conceptual samples such as examples, tutorial and quizzes to assess themselves, and they can participate in collaborative activities and work in groups. To develop the tool design, an analysis of all requirements is required the details of analysis, design, implementation and testing explained. The tool development model used during the stage of analysis, design stage and implementation comprise the rational unified process. RUP is a new software process model that is derived from Unified Modeling Language (UML). Rational Unified Process is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organisation. RUP’s goal is to ensure the production of high-quality software that meets the needs of its end-users within a predictable schedule and budget. RUP consist of four phases, as illustrated in Figure 3.2.
### 3.3.1 Inception

During the inception phase, the business case for the system is established and the project scope is defined. To accomplish this, all external entities must be defined as well, such as who will interact with the system (actors) and define the nature of this interaction at a high level. The main outcomes of the inception phase are the purpose, scope and objectives, assumptions and constraints of the project plan. These outcomes are defined in chapter 1.

![Figure 3.2: Rational Unified Processes](image)

### 3.3.2 Elaboration Phase

The purpose of the elaboration phase is to analyse the problem domain and establish a sound architectural foundation. Architectural decisions have to be made with an understanding of the whole system that includes designing teaching framework and analyzing major functionality and
nonfunctional requirements based on that framework. The main outcome of this phase, as far as this research is concerned, are firstly the teaching framework; and secondly, the use-case models. Teaching framework is for making software engineering students to have a deep understanding of software inspection and to improve their practical abilities. The framework consists of three parts: general guidelines, specific guidelines and learning activity. In addition, there are two parts in general guidelines, namely software inspection concept and software inspection technique. Specific guidelines include software inspection process and applying the techniques on software inspection. The third layer includes conceptual samples, collaboration in practice, work samples and assessments for improving practical skills and abilities.

Second outcome is Use-Case model (at least 80% complete). All use cases and actors have been identified, and most use-case descriptions have been developed. The third outcome is the software architecture description, which includes the supplementary requirements. The supplementary requirements capture the nonfunctional requirements which describe the candidate’s nonfunctional requirement needs to be derived from such systems based on the architectures. This phase highlights requirements, analysis and design. In this phase, the gathered requirements are analysed. Analysis concerns a detailed study of existing sources and software teaching tools, which leads to discover the specification of the new system. There are many representation forms that are used in the software design method. These forms including textual and diagrammatical of notation, such as data flow diagram, entity relation diagram and UML. In this research the analysis design model is proposed using the UML model. The details will be described in Chapter 4.
3.3.3 Construction Phase

During the construction phase, all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. This phase has aim to build the teaching tool and the released versions that emerge as a result of the implementation steps are described. Hence, the system that is the result of this research is described here. The system satisfies the main the WBTSIS tool, is to provide a learning environment for students to learn software inspection concepts and techniques step by step. In addition, it provides the ability to guide software engineering students to perform conceptual samples for assessing themselves or to execute the software inspection process and its techniques collaboratively in a group. This tool will be developed using a visual studio (C#.net). SQL server 2008 is used as the relational database. The detail will be described in Chapter 5.

3.4. Evaluation

The purpose of the transition phase is to have the software product evaluated by the user community. Once the product has been given to the end user, issues usually arise that require the development of new releases, correction of some problems, or completion of the features that were postponed. The main outcomes of the WBTSIS tool are to illustrate all concepts of software inspection with definition, examination and tutorial, and to guide the software engineering students to individually or collaboratively perform of the tasks and specifying techniques within case studies or projects. A set of questionnaires is designed as a means of evaluating the system. An evaluation form is used to get feedback from users (software engineering students) on while
the WBTSIS teaching tool covers all functionalities and mentioned outcomes. The gathered data is analysed to illustrate and to evaluate the usability and functionality of this system.

3.5 Summary

This chapter provides an overview of the methodology used in this research. Firstly, the research explains about software inspection and its techniques that use for detecting defects. The contribution of the research to the teaching of software inspection as a web-based as well as the addition of a new property for teaching, practicing and collaboration are also described. Secondly, this chapter outlines the Rational Unified Process (RUP) as software process modelling used in this research together with its iterative behaviours for the deliverables of each phase in this research.
CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter describes the system analysis and design for WBTSIS. This analysis and design are derived based on goals and approaches as discussed in Section 3.2. UML diagrams are used to present the analysis and design of WBTSIS.

4.2 System Overview

WBTSIS is the tool to support software engineering student for learning software inspection and its techniques. The system will help student to perform software inspection concepts step-by-step in learning activity such as tutorial, quizzes and work samples.

4.3 Teaching Framework

A teaching framework is offered as a tool to enable software Engineering students have a deep understanding of software inspection and to improve their practical abilities in university and career. The main goal for presenting this framework is to guide software engineering students to overcome the complexity of software inspection knowledge and to obtain appropriate learning skills and practical abilities. The framework consists of three parts: general guidelines, specific guidelines and learning activity. In addition, there are two parts in general guidelines namely, Software inspection concept and software inspection techniques. Specific guidelines include Software inspection process and applying the techniques on software inspection process.
The third layer includes conceptual samples, collaboration in practice and work samples for improving practical skills and abilities. The framework will be explained in detail with illustration in the following pages.

4.3.1 General Guidelines

The teaching tool is presented to support students learning about software inspection. It enhances the efficiency of software inspection learning because understanding the subject is difficult due to the complexity of its concepts and its techniques. This framework helps software engineering students to learn about when, where and why to use the knowledge they are learning. The section on general guidelines is designed with two components, software inspection concept and principle, and software inspection techniques, as shown in Figure 4.1.
4.3.1.1 Software Inspection Concept and Principle

In order to effective teach software, ambiguous and difficult definitions must be avoided and a basic explanation of software inspection given. These concepts and definitions should be taught to software engineering students using a combination of multimedia method such as combining texts, sounds, graphics, images, animation and interactive features, as it is the best way to achieve efficiency and high quality of teaching. This teaching methods improves efficiency in
teaching and motivation software engineering students to learn, be more interested in thinking software inspection, and be eager to answer questions such as: why Software Inspection is important; when can software inspection be used; and how software inspection should be used. In this section, students will achieve primary knowledge of software inspection and its concepts.

4.3.1.2 Software Inspection Technique

It is imperative that students well understand software inspection technique because it is the key activity for defect detection. Such techniques are referred to as reading techniques. After learning about techniques, students will be more interested in thinking about software inspection and eager to apply the techniques for software inspection. The techniques will help students to identify defects in software documents which will increase the effectiveness of an inspection team (Laitenberger and DeBaud, 2000). The most popular reading techniques are ad-hoc reading and checklist-based reading.

4.3.2 Specific Guidelines

This section can be considered as the main part for students’ understanding and consist of two components, software inspection process and roles and applying technique on Software inspection process. A more specific focus is given to this section on finding defects and selecting appropriate techniques so that students will be able to efficiently find defects and execute tasks using specified techniques.
4.3.2.1 Software Inspection Process and Roles

Using the general guidelines, students will become familiar with the primary Software inspection concept. In this part, the software inspection process is presented step by step using teaching tool help make understanding easier. This component of general guidelines teaches students about team roles and team size for the software inspection process. This component also indicates how many persons are assigned to each role and how to select people for each role. The next step involves each person doing his/her task in software inspection process based on his/her specified role, such as analysing the document, identifying defects and correcting them. Some students may face many problems and have question about this part. To solve students’ problems, this part should be taught step-by-step. Supporting learning by students who have different level of understanding and knowledge of software inspection, from the easy stages to the more difficult.

4.3.2.2 Applying technique on Software Inspection Process

Techniques are used for supporting defect detection in a requirement document and may increase the effectiveness of an inspection team. The teaching tool demonstrates how a technique can be chosen and how it can be applied on particular document based on this framework. The techniques help a student to scrutinise a document for identify defects. Finally, students can answer questions how techniques can be applied to software inspection and which technique is useful. This part is very critical and difficult to learn because identifying techniques is not easy and needs to more concentration, background information and skill.
4.3.3 Learning Activity

For software engineering students to learn effectively, the learning activity has to be practical as it means that there are more opportunities to practise. There are a lot of samples to evaluate students’ practical abilities about the software inspection process. The main goal of this part is to build a flexible model on software inspection and its dimensions by using a lot of conceptual and work samples, as well as collaboration with other students. This tool prepares a way for both experts and novices to improve their knowledge and practical abilities. Learning activities help students to improve their understanding and abilities about concepts and practice, whether in groups or as individuals and students can also evaluate their obtained knowledge. Students can practise based on knowledge that they gained from all guidelines and concept approaches. This framework improves software engineering students’ creativity and initiative for executing tasks. Finally, they will gain experience for their careers and research in the industry.

4.3.3.1 Conceptual Samples:

Using conceptual samples is an effective way to present all the concepts of software inspection. Conceptual samples include examples, tutorials and quizzes. They show students how they can gain software inspection knowledge and where they can use it. The method prepares students for the practical step of learning as students will execute all tasks individually or in groups. Quizzes help to evaluate the level of students’ understanding about software inspection concepts. It is a good method for a teacher to evaluate students. Sometimes, it encourages students to learn better and to focus more on a concept and principle. They can also improve their
learning skills. Students’ thinking should be clearly elicited by using quizzes and feedback must be provided.

4.3.3.2 Work Sample

Project-based learning samples or case studies are needed to teach student how to identify defects. Students should do inspection processes on the case study or project individually or in groups. Students should analyse document, identify defect, discuss about defects and correct them by specified techniques. If they face problems, they can refer to the guidelines and get help. They can send the results to the teacher for correction and get feedback.

4.3.3.3 Collaboration in Practice

Collaborative learning helps students to learn more effectively. Many educators place a high premium on teaching strategies that go beyond more mastery of content and ideas. Students collaborate with each other in group to learn the dynamics of working with each other and to take role for identifying and correcting faults in the document samples. Each group consists of five or ten persons. The member of these small groups chooses their special roles based on their knowledge and experience in those roles. Every group should finish their work on time before another task start. If a group member faces a problem and is unable to solve it, she/he can seek help from another group. The students do their tasks and create solutions according to their roles and responsibilities and finish process collaboratively. Extending this area of concern to professionals leads to increased project-based and team-based activities that are grounded on sharing of tasks and responsibilities. The end goal is to lessen individual competition that has been well entrenched in almost all working environments in different canters of learning and professional endeavours.
4.4 System Requirement

4.4.1 Functional Requirements Specification

This part concentrates on the Software Inspection teaching tool analysis from all aspects by providing the various diagrams that exhibit the analysis of this application. This research is using Object Oriented Analysis to analyze the system. The first one is listing out the functional and non-functional requirements as well as the construction rules. The second one is about behavioral analyzing of the system, which means analyzing how the system functionality interacts internally in terms of Use Case diagrams and the class diagram are investigated and illustrated to capture the structure of the system.

Table 4.1: Functional Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Functional Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>WBTSIS shall allow to student to sign in with user name and password.</td>
</tr>
<tr>
<td>REQ2</td>
<td>WBTSIS shall allow to student to access general guidelines.</td>
</tr>
<tr>
<td>REQ3</td>
<td>WBTSIS shall allow to student to view software inspection concepts.</td>
</tr>
<tr>
<td>REQ4</td>
<td>WBTSIS shall allow to student to view definition of software inspection techniques.</td>
</tr>
<tr>
<td>REQ5</td>
<td>WBTSIS shall allow to student to access specific guidelines.</td>
</tr>
<tr>
<td>REQ6</td>
<td>WBTSIS shall allow to student to view definition of software inspection processes.</td>
</tr>
<tr>
<td>REQ7</td>
<td>WBTSIS shall allow to student to view applying techniques on software inspection process.</td>
</tr>
<tr>
<td>REQ8</td>
<td>WBTSIS shall allow to student to access conceptual samples.</td>
</tr>
<tr>
<td>REQ</td>
<td>WBTSIS shall allow to student/teacher to perform the following activities.</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REQ9</td>
<td>WBTSIS shall allow to student to view examples.</td>
</tr>
<tr>
<td>REQ10</td>
<td>WBTSIS shall allow to student to do tutorials.</td>
</tr>
<tr>
<td>REQ11</td>
<td>WBTSIS shall allow to student to take part on quizzes.</td>
</tr>
<tr>
<td>REQ12</td>
<td>WBTSIS shall allow to student to access to quizzes’ answers.</td>
</tr>
<tr>
<td>REQ13</td>
<td>WBTSIS shall allow to student to access work samples.</td>
</tr>
<tr>
<td>REQ14</td>
<td>WBTSIS shall allow to student to send result to teacher.</td>
</tr>
<tr>
<td>REQ15</td>
<td>WBTSIS shall allow to user to attend collaboration in practice.</td>
</tr>
<tr>
<td>REQ17</td>
<td>WBTSIS shall allow to student to put on group.</td>
</tr>
<tr>
<td>REQ18</td>
<td>WBTSIS shall allow to student/teacher to view group’s progress.</td>
</tr>
<tr>
<td>REQ19</td>
<td>WBTSIS shall allow to user to search for finding desired information.</td>
</tr>
<tr>
<td>REQ20</td>
<td>WBTSIS shall allow to student/teacher to view group’s progress.</td>
</tr>
<tr>
<td>REQ21</td>
<td>WBTSIS shall allow to teacher to create general guidelines.</td>
</tr>
<tr>
<td>REQ22</td>
<td>WBTSIS shall allow to teacher to modify general guidelines.</td>
</tr>
<tr>
<td>REQ23</td>
<td>WBTSIS shall allow to teacher to create specific guidelines.</td>
</tr>
<tr>
<td>REQ24</td>
<td>WBTSIS shall allow to teacher to modify specific guidelines.</td>
</tr>
<tr>
<td>REQ25</td>
<td>WBTSIS shall allow to teacher to create conceptual samples.</td>
</tr>
<tr>
<td>REQ26</td>
<td>WBTSIS shall allow to teacher to modify conceptual samples.</td>
</tr>
<tr>
<td>REQ27</td>
<td>WBTSIS shall allow to teacher to create work samples.</td>
</tr>
<tr>
<td>REQ28</td>
<td>WBTSIS shall allow to teacher to modify work samples</td>
</tr>
<tr>
<td>REQ29</td>
<td>WBTSIS shall allow to teacher to create group.</td>
</tr>
<tr>
<td>REQ30</td>
<td>WBTSIS shall allow to teacher to put student to group work.</td>
</tr>
</tbody>
</table>
4.4.2 Non-functional Requirements Specification

Main non-functional requirements of WBTSIS are given Table 4.2.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Non-Functional Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usability</td>
<td>- The system should be easy to use and follow.</td>
</tr>
<tr>
<td></td>
<td>- Language used in the system should be understood unambiguously by non-technical users.</td>
</tr>
<tr>
<td></td>
<td>- Navigation should be fast and easy.</td>
</tr>
<tr>
<td>Operational</td>
<td>- The system must be able to be accessed from anywhere.</td>
</tr>
<tr>
<td>Performance</td>
<td>- The system must be able to process high load of data.</td>
</tr>
<tr>
<td></td>
<td>- Lecture notes and other data must be stored in archive for future reference.</td>
</tr>
<tr>
<td></td>
<td>- The system must be user friendly, and easy to use for all users, especially for students.</td>
</tr>
<tr>
<td></td>
<td>- The system can accommodate up to 100 users and respondents at a time. (Scalability test is performed using Microsoft application Centre Test, which is provided in the Visual studio.Net 2005 package)</td>
</tr>
<tr>
<td>supportability</td>
<td>- The system should be easy to be modified and upgraded to accommodate future enhancements.</td>
</tr>
<tr>
<td>Security</td>
<td>- All users of the system will require authentication to use System.</td>
</tr>
<tr>
<td>Portability</td>
<td>- The system shall be accessible through multiple web browser platforms</td>
</tr>
</tbody>
</table>
### 4.4.3 Actors and Roles

#### Table 4.3 Actors & Roles

<table>
<thead>
<tr>
<th>Actor name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Teacher</strong></td>
<td>A person who uses the system and responsibility such as registration, creating/modifying guidelines, creating/modifying conceptual samples, creating/modifying work samples and put students to different groups.</td>
</tr>
<tr>
<td><strong>Student</strong></td>
<td>A person who can register on system and login to access information.</td>
</tr>
<tr>
<td><strong>Learner</strong></td>
<td>A person who can do all learning activity such as examples, tutorials, quizzes and group projects. She/he can access to general and specific guidelines for better understanding about inspections.</td>
</tr>
</tbody>
</table>
4.4.4 Use case

Use case diagram is one of the most important UML diagrams, which describes the system main functional requirement in the form of “use case”, the system’s actor who interact with those use cases and relationship between all of the system’s use cases and actors. The use case diagrams for this teaching tool are shown in below.

Figure 4.2 Use Case Diagram
4.4.5 Use Case Description

In this section each use case in the use case diagram is explained in the use case description as detailed. Use case description makes clears the preconditions, post conditions, actions, alternative and Scenarios for each use case. Table 4.4 till Table 4.12 show the use case descriptions for the WBTSIS.

UC1. Login Operation

Table 4.4: WBTSIS Use Case for Login Operation

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td><strong>Post condition :</strong></td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Alternative Paths</strong></td>
</tr>
<tr>
<td><strong>Exception condition</strong></td>
</tr>
</tbody>
</table>
UC2. Manage General Guidelines

Table 4.5: WBTSIS Use Case for Manage General Guidelines

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td><strong>Post condition:</strong></td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td>1. If teacher chooses to create software inspection</td>
</tr>
<tr>
<td>2. If teacher chooses to modify software inspection</td>
</tr>
<tr>
<td>3. If teacher chooses to create software inspection techniques</td>
</tr>
<tr>
<td>4. If teacher chooses to modify software inspection techniques</td>
</tr>
<tr>
<td><strong>Exception condition</strong></td>
</tr>
</tbody>
</table>
Table 4.6: WBTSIS Use Case for Create Software Inspection Concepts

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td><strong>Post condition:</strong></td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Exception condition</strong></td>
</tr>
</tbody>
</table>
## Modify Software Inspection Concepts

### Table 4.7: WBTSIS Use Case for Modify Software Inspection Concepts

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case name</td>
</tr>
<tr>
<td>Brief description</td>
</tr>
<tr>
<td>Actors:</td>
</tr>
<tr>
<td>Precondition:</td>
</tr>
<tr>
<td>Post condition :</td>
</tr>
<tr>
<td>Flow of event</td>
</tr>
</tbody>
</table>

1. From main page, teacher clicks on Manage General Guidelines and then chooses “Modify software inspection Concepts”.

2. The system will display a form to choose whether you want to modify new information about software inspections and their techniques.

   2.1. Teacher clicks “save changes”
   2.2. Teacher clicks “delete information”

3. System update/delete information about software inspection and system displays successfully.
### UC5. Manage Conceptual Samples

Table 4.8: WBTSIS Use Case for Manage Conceptual Samples

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td>Manage Conceptual Samples</td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td>Teacher provides managing Conceptual Samples.</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td>teacher</td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td>WBTSIS tool is loaded and the login operation is performing correctly.</td>
</tr>
<tr>
<td><strong>Post condition:</strong></td>
</tr>
<tr>
<td>The conceptual samples are displayed to the students and saved/update in database.</td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td><strong>Actor</strong></td>
</tr>
<tr>
<td>1. If teacher chooses to create examples</td>
</tr>
<tr>
<td>a. Include Create examples</td>
</tr>
<tr>
<td>2. If teacher chooses to modify examples</td>
</tr>
<tr>
<td>a. Include modify examples</td>
</tr>
<tr>
<td>3. If teacher chooses to create tutorials</td>
</tr>
<tr>
<td>a. Include create tutorials</td>
</tr>
<tr>
<td>4. If teacher chooses to modify tutorials</td>
</tr>
<tr>
<td>a. Include modify tutorials</td>
</tr>
<tr>
<td><strong>Exception condition</strong></td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>
UC6. Create Quizzes

Table 4.9: WBTSIS Use Case for Create Quizzes

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case name</td>
</tr>
<tr>
<td>Brief description</td>
</tr>
<tr>
<td>Actors:</td>
</tr>
<tr>
<td>Precondition:</td>
</tr>
<tr>
<td>Post condition:</td>
</tr>
<tr>
<td>Flow of event Actor</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exception condition</td>
</tr>
</tbody>
</table>
UC7. Use General Guidelines

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td>1. From main page, student can choose “software inspection concepts/ software inspection techniques” for main menu.</td>
</tr>
<tr>
<td>2. The system will display a form of software inspection concepts/ software inspection concepts.</td>
</tr>
</tbody>
</table>
### UC8. Use specific Guidelines

#### Table 4.11: WBTSIS Use Case for Use specific Guidelines

<table>
<thead>
<tr>
<th>Use Case ID: USE CASE 8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use case name</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
</tr>
<tr>
<td><strong>Precondition:</strong></td>
</tr>
<tr>
<td><strong>Flow of event</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
4.5 Class Diagram

Class diagram consists of objects represented by rectangular shapes divided into three parts; namely the name of the class, the collection of attributes, the collection of candidate operations. The Class diagram is used for modeling the objects and classes that collaborate with each other to achieve the behavioral way that is already captured in a use case model.

Figure 4.3 provides an idea about the enhancement of the abstract class diagram in terms of refining the association relationships into inheritance, compositions or aggregations. From the diagram below, we can find that the relation among Users, Admin and System Analyst classes are modified into inheritance relation as well as the relation among Concerns, Logical Concerns and Physical Concerns. Also we observe that the candidate attributes and operations are added to the detailed class diagram.
Figure 4.3 Class Diagram
4.6 Sequence Diagram

Sequence diagram describes the dynamic aspect of a system. Sequence diagram shows the sequence of messages sent between the objects with respect to time. The sequence diagram captures the behavior of a single use case and show interaction pattern among the object. In the following, a few of WBTSIS sequences diagram are shown.

In the following, one of WBTSIS sequence diagrams is shown in Figure 4.10 while the remaining of the sequences diagrams can be referred to in Appendix A.
**Scenario: Manage General Guidelines**

1. Teacher login to system
2. Choose general guidelines
3. Choose software inspection concept from main menu
4. Create software inspection concepts
5. Add new information to database

![General Guidelines Sequence Diagram](image)

*Figure 4.4 General Guidelines Sequence Diagram*
4.7 Summary

This chapter investigates into the analysis phase of this project. The first section discussed the functional and non-functional requirements in terms of listing them out as well as the construction rules for concerns, relationships and predicate constraint. The second section was about the environment modeling using context diagram or block diagram that is used to show the environment where the system will work. The third section considered use cases diagrams for the system as a whole that capture the functional requirement for the system. The fourth section described class diagram. Lastly, sequence diagram was designed.
CHAPTER 5

WBTSIS IMPLEMENTATION

5.1 Introduction

This chapter describes the implementation of the WBTSIS tool. In the previous chapter, all the requirements required for developing the WBTSIS tool were elicited and analyzed. In this chapter, based on gathered information, the detailed design of WBTSIS is represented by using different object-oriented modelling diagrams. The architecture and components of WBTSIS are also explained to clarify implementation of the tool. The most important screen shots of WBTSIS are shown to display the design of the user interface of the tool. Finally, software and hardware requirements needed for development of WBTSIS are specified in the table.

5.2 Constructing WBTSIS Based on Teaching Framework

This section discusses the teaching framework which affects the WBTSIS tool. There is a particular teaching framework behind the WBTSIS that enables it to analyse requirements. This framework is explained in more detail in Chapter 4.

5.3 Architecture of WBTSIS Tool

5.3.1 Client server architecture

A client server is the network server which separates a client from a server. Each instance of client software can send requests to servers. Specific types of server include application servers, file servers and mail servers. Users send the requests to the web server and the web server finds
all the information through the database server. The web server will send the information back to web browser to let the user look at it. One of the types of client server architecture is three – tier client server architecture. This three-tier client server architecture has been chosen for WBTSIS development. The system concept is suitable for the client server concepts as WBTSIS is a web-based system that needs to be accessed through the internet. The architecture is suitable and provides a scalable architecture. The three–tier client server architecture is divided in to three phases which are:

5.3.1.1 Client Tier

The client manages the user interface where the processes of the user take place. The client for WBTSIS is the computer used by teacher and students.

![Figure 6. 11 Three-tier Architecture Model (Juel, et al., 2005)](image-url)
5.3.1.2 Application Servers/Tier

The application server is used to process all the business and data processing logic for the clients in which a server computer is dedicated for running certain software applications. Here, the systems will perform query/update processing and then transmit responses to users. WBTSIS will process all the collaborative and individual activities performed by all parties. This application supports file uploading and downloading, real-time message transfers, online editing and e-mailing.

5.3.1.3 Database Servers/Tier

The database server will generate data validation and database requests for transmitting to the server. The database provides the database services to other computer programmes or computers. Here, the server will accept and process the database requests from clients and check the authorisation. The database management system used in WBTSIS is Microsoft SQL Server 2008 which frequently provides database server functionality. The database keeps all the students’ and teachers’ records as well as transactions made by them.
5.4 Design of WBTSIS tool

5.4.1 Component Diagram

![Component Diagram for WBTSIS](image.png)

Figure 5.12 Component Diagram For WBTSIS
In a software development project, there will be many files that make up the system. These files will have dependencies on one another. Component diagram as is one of the two types of implementation diagrams in Unified Modelling Language (UML) (Bennet et al., 2002). As illustrated by Figure 5.2, this component diagrams represent the dependencies of the components on some applications, with or without the use of the interface.

WBTSIS has few major interfaces that consist of data access to different types of components which are registration, teacher, and student. Each of them is meant for a different purpose even though some of them may be used for the same activities. The “registration” component is used for administration while the “teacher” will manage the general and specific guidelines and activities. Both “teacher” and “student” components have data access to software inspection notes and conceptual samples as well as work samples. These components will also be engage in the collaborative application. However, each of these components is protected by the security access control whereby the encryption is used for login purposes. In another words, all are protected by the security infrastructure to ensure the security and access level of the system. The persistent infrastructure is used for determining a system, where the system can determine the appropriate response according to the input by the components. Principally, WBTSIS is provided with a database that uses Microsoft SQL Server 2008 to handle all the components, applications and infrastructure in terms of data storage and retrieval.

5.4.2 Package Diagram

Package diagrams ensure that developers organise model elements into groups, making it simpler and easier to understand other UML diagrams. Packages are depicted as use-case
diagrams and class diagrams because these models have a tendency to grow. WBTSIS contains the following packages:

- Database package which consists of classes for connecting to the database of system.
- Database Services package which consists of classes for calling database services.
- ASP.net Classes package which consists of ASP.net classes of the WBTSIS tool as explained in the object model section in the previous chapter of this research.
- ASPX files package which includes ASPX files which construct the WBTSIS interface.

Figure 5.3 visualizes the package diagram of WBTSIS.

![Package Diagram of WBTSIS]

Figure 6.13 Package Diagram of WBTSIS
5.5 User Interface Design

5.5.1 Overview of the Current System

WBTSIS is a web-based system to teaching software inspection processes and techniques to software engineering students. This system can be accessed via the internet at Faculty of Computer Science and Information Technology (FSKTM) within the time frame set by the teacher. WBTSIS was developed using Active Server Pages (ASP) and C# programming language. This system integrates with the students’ database known as Integrated Student Information System (ISIS). The database used in WBTSIS was Microsoft SQL Server 2008 and the Web server used was Internet Information Services (IIS). Overall, the software development of this system is a proprietary software. WBTSIS is divided into four main modules. The said modules are general guidelines, specific guidelines, conceptual work and work samples.

5.5.2 WBTSIS Execution

This section of the report explains how the WBTSIS works and how the software inspection process is conducted. At the preparatory step, the lecturer creates the general guidelines, describes it that include concepts of software inspection and its techniques. Also during this stage, the lecturer also provides some appropriate and related resources for more information. Furthermore during this preparatory step, the lecturer creates specific guidelines and explains the software inspection process and how to apply techniques on software inspection. To support this task WBTSIS displays the information containing these subjects and related references which
enables the user to search and find specific topics. In this case, students can become familiar with the topic and use the e-mail to ask for clarification in case there are some doubts. After everything is clarified, the lecturer can use the “conceptual samples” to take everyone to the next stage.

The purpose of this step is to create an environment for practicing and learning more about software inspection and its techniques. The lecture creates some examples, tutorials and quizzes with prepared answers to motivate students. This means that each learner can read the examples, do tutorials and take quizzes and check the answers. Ideas created during this stage are only visible to the students for the purpose of learning, practising and improving their knowledge. This step also prepares students for the next step.

Software inspection includes many processes and techniques, one of which is idea creation which involves discussion and clarification of the software inspection process and techniques. The idea creation step enables that student to perform the software inspection process in a group and with real projects. The lecturer puts learners in different groups and gives them different responsibilities. During this step participants can also communicate with others as well as they will use the chat windows. The system allows the lecturer to view the status of each participant. This is the end of this step and the lecturer can view a report of the learner.
5.5.3 WBTSIS Interface

The design of the user interface was guided by human computer interaction (HCI) principles in order to improve the usability of the WBTSIS tool. The interface consists of three main views: general guidelines, specific guidelines and samples activity. The design of these views applies the tab metaphor, in order to improve the familiarity and visibility of the interface, and makes the navigation between its parts easier. The screenshots of the tool’s interface (illustrated in the subsequent figures) show the consistency of the design between the tasks. The following illustrations represent the main GUI (Graphical User Interface) page of WBTSIS.
The figure below shows the input design interface for the registration of a new student. Students can navigate easy and fast and enter their information rapidly.

![Student Registration Interface](image)

Figure 5.15 Sign up
In the login page, both student and teacher can login to system. If students’ username and password was not correct, system shows error massage.

Figure 5. 16 Login
In this following Figure represents that student can select desire topic and visit related information such as its concepts, samples, etc. In Figure 5.8, Student can do her/his activity. Since the objective of this tool is to teach practically student the concept of software inspection.
If students choose view concepts, the new page will be open and show related information. In figure below presents the desired information. Also student can click on video or related link to more information.
Figure 5. 19 Student's Selection

If student click on group project, he following page open and student can register for participant on inspection group. In this page, level of software inspection is shown.
IBIS is a web-based tool to support geographically dispersed inspection teams. Based on findings from empirical studies of software inspections, the IBIS tool adopts a reorganization of the inspection process to minimize synchronous activities and coordination problems. The adopted inspection process includes well-defined roles to be assigned to each member of the inspection team and seven consecutive stages to be followed during the process.

**Inspection Team**

The inspection team is a group of peer staff members with a special interest in the product to be inspected. The minimum team size is three persons, although a team might scale up to an unlimited number (in a traditional inspection process the typical team size may vary between three and seven). The roles in an inspection team are:

- Inspector
- Leader
- Author
- Moderator

**Figure 8. Work Samples**
5.5.4 Tool Implementation

5.5.4.1 Development of WBTSIS

WBTSIS is a web based application which it runs on WWW protocol. There are several applications that have been used for development of web-based systems like WBTSIS. In the following, some important components needed for WBTSIS development are described.

i. Database Server

Database server is an important application which is dedicated to database storage and retrieval that belongs to the WBTSIS. Microsoft SQL Server 2008 is the database server for developing WBTSIS.

ii. Development Language

Source codes of WBTSIS are written with C# language. C# is a powerful and flexible language that it is suitable for developing web-based applications. Therefore using C# language, all of source codes required for developing WBTSIS are written.

5.5. Summary

This chapter discussed the different stages of developing the WBTSIS tool, which implements the proposed teaching software inspection. WBTSIS’s main functionalities are to
provide the software engineer students how to learn with difficult concept of software inspection by using learning theories and how to work and collaborate with each other easy and fast.
CHAPTER 6

TESTING AND EVALUATION

6.1 Introduction

After coding, a programmer must test each program to make sure that it functions correctly (Shelly, 2006). In order to identify and eliminate execution and logic errors, WBTSIS was tested using both system-testing and user-evaluation. The first part of this chapter will cover the testing of WBTSIS using various testing techniques and test cases. The second part of this chapter describes the evaluation process of WBTSIS and its results. The results include the summary of the participants’ backgrounds, the aspects of WBTSIS evaluation and the suggestions given by users.

6.2 WBTSIS Testing

Software testing is an important activity in every software development environment. It is a critical phase in software quality assurance which ensures that the system complies with its requirements and specifications. The intention is to test the system’s ability and to discover errors. Before the system is implemented, testing of the system must be done to ensure that there are no problems. WBTSIS has undergone three stages of testing, which are: Unit Testing, Integration Testing, and System Testing. These three testing stages are discussed in the following paragraphs.
6.2.1 Unit Testing

From its name, unit testing is defined as the process of testing individual software components. It is performed for checking several functionalities and services of the WBTSIS tool through the source code, such as all classes inside the WBTSIS. In actual fact, each implemented unit of the system is tested during the process of unit testing. By debugging and compiling each unit of source code, all types of errors will be recognized and eliminated.

6.2.1.1 Functions to be tested

To ensure that all functions tested, a checklist has been prepared. Table 6-1 is the checklist which has been duly prepared for testing of the system.

<table>
<thead>
<tr>
<th>Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Information</td>
<td>• View User Information&lt;br&gt;• Add User Information&lt;br&gt;• Delete User Information</td>
</tr>
<tr>
<td>General Guidelines</td>
<td>• Add Software Inspection Concepts&lt;br&gt;• Delete Software Inspection Concepts&lt;br&gt;• Update Software Inspection Concepts&lt;br&gt;• Add Software Inspection Techniques&lt;br&gt;• Delete Software Inspection Techniques&lt;br&gt;• Update Software Inspection Techniques</td>
</tr>
<tr>
<td>Specific Guidelines</td>
<td>• Add Software Inspection Process&lt;br&gt;• Delete Software Inspection Process&lt;br&gt;• Update Software Inspection Process&lt;br&gt;• Add Applying Techniques on Software Inspection&lt;br&gt;• Delete Applying Techniques on Software Inspection&lt;br&gt;• Update Applying Techniques on Software Inspection</td>
</tr>
</tbody>
</table>
6.1.1.2 Validation Function

This section explains in detail the unit testing of form validation function for all screens in this system. Table 6-2 shows the test steps taken and the expected result. The bugs are corrected and the pages retested until they work as intended.

<table>
<thead>
<tr>
<th>Id</th>
<th>Test step</th>
<th>Expected Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Username or password incorrect or empty.</td>
<td>Error: a. Login ID or Password is incorrect. Please try again.</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>Add new user account:</td>
<td>Error/Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ii. Record exists</td>
<td>Error: a. Record Exists. Please key in unique User Account.</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Update user account:</td>
<td>Message: a. Record updated</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Delete record</td>
<td>Message: a. Are you sure you want to delete record?</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update user account:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Record updated</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Add General guidelines</td>
<td>Error/Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i. Did not key in required field.</td>
<td>Error:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Please key in general guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Please key in general guidelines</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ii. Record exists</td>
<td>a. Record Exists. Please key in unique general guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update record</td>
<td>Message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Record updated</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete record</td>
<td>Message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Are you sure you want to delete record?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Record deleted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Add Specific guidelines</td>
<td>Error/Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i. Did not key in required field.</td>
<td>Error:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Please key in specific guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Please key in specific guidelines</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ii. Record exists</td>
<td>a. Record Exists. Please key in unique specific guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update record</td>
<td>Message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Record updated</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete record</td>
<td>Message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Are you sure you want to delete record?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Record deleted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Add Specific guidelines</td>
<td>Error/Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i. Did not key in required field.</td>
<td>Error:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Please key in specific guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Please key in specific guidelines</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ii. Record exists</td>
<td>a. Record Exists. Please key in unique specific guidelines Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Update record** | Message:  
| a. Record updated | OK |
| **Delete record** | Message:  
| a. Are you sure you want to delete record?  
b. Record deleted | OK |
| **6 Add conceptual samples** | Error/Message |
| i. Did not key in required field. | Error:  
| a. Please key in conceptual samples Code  
b. Please key in conceptual samples Description | OK |
| ii. Record exists | a. Record Exists. Please key in unique conceptual samples Code | OK |
| **Update record** | Message:  
| a. Record updated | OK |
| **Delete record** | Message:  
| a. Are you sure you want to delete record?  
b. Record deleted | OK |
| **7 Add work samples** | Error/Message |
| i. Did not key in required field. | Error:  
| a. Please key in work samples Code  
b. Please key in work samples Description | OK |
| ii. Record exists | a. Record Exists. Please key in unique work samples Code | OK |
| **Update record** | Message:  
| a. Record updated | OK |
| **Delete record** | Message:  
| a. Are you sure you want to delete record?  
b. Record deleted | OK |

### 6.2.2 Integration Testing

Integration testing is the phase of the software testing during which individual software modules are combined and tested as a group. Integration testing is conducted to test the newly developed features with the existing features. For WBTSIS, the working units or modules which
are tested during the unit testing are integrated with the newly developed module and tested together in a bottom-up approach.

Table 6.14: Test case for Registration

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register in system</td>
<td>Create an account</td>
<td>System must display successful message</td>
<td>System creates new account for user and</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>and click Sign up button</td>
<td>and create new account into database</td>
<td>displays successful message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create Software Inspection Concepts</td>
<td>System creates Software Inspection Concepts and upload video successfully and displays successful message.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.15: Test case for Create Software Inspection Concepts

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Software Inspection Concepts</td>
<td>Create information related to software inspection or upload video And click create button</td>
<td>System must display successful message and add new information about software inspection into database</td>
<td>System creates Software Inspection Concepts and upload video successfully and displays successful message.</td>
<td>Pass</td>
</tr>
</tbody>
</table>
### Table 6.16: Test case for Update Software Inspection Concepts

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Software Inspection Concepts</td>
<td>Change information related to software inspection</td>
<td>System must display successful message and change information about software inspection into database</td>
<td>System update Software Inspection Concepts successfully and displays successful message.</td>
<td>Pass</td>
</tr>
<tr>
<td>And click upload button</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.17: Test case for Create Quiz

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Quiz</td>
<td>Upload quiz and its answer and click create quiz button</td>
<td>System must display successful message and add quiz &amp; answer into database</td>
<td>System create quiz and displays successful message</td>
<td>Pass</td>
</tr>
</tbody>
</table>

| | | | | |
Table 6.18: Test case for Create Project

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Project</td>
<td>Enter project detail and click on Create Project button</td>
<td>System must display successful message and create new project into database</td>
<td>System creates new project and displays successful message</td>
<td>Pass</td>
</tr>
</tbody>
</table>

Table 6.19: Test case for View Group Practice

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Items</th>
<th>Expected Results</th>
<th>Test Results</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Group Practice</td>
<td>Select project and then View group member</td>
<td>System must display list of member which belong to the selected project</td>
<td>System displays user information on selected project from database</td>
<td>Pass</td>
</tr>
</tbody>
</table>

6.2.3 System Testing

System testing is the last phase of the WBTSIS testing stages. In this stage, WBTSIS focuses more on testing the behaviour of the entire system rather than on the functional aspects. The emphasis is on the system performance and the data integrity of the system.
6.3 WBTSIS Evaluation and Results

After performing the above mentioned types of testing on the system, an evaluation of the developed system is carried out from view points of the users. The purpose of evaluation is to guarantee that the developed system fulfills its requirements which were agreed upon in the previous stages.

6.3.1 Evaluation Objectives

The objectives of this evaluating the WBTSIS are:

• To assess the teaching framework
• To identify the system’s ease of use.
• To assess the system’s functionalities.

6.3.2 Participant

The evaluation of the system involves 15 university students from the Faculty of Computer Sciences of University Malaya. The participants of the evaluation are all master students at the Computer Science department and as such are familiar with computers. The 15 participants were divided into three groups consisting of five members per group. Each group includes different students with different experience and knowledge.

6.3.3 Methodology of Evaluation

This section describes the methodology used to conduct the evaluation. The evaluation was started by distributing an evaluation’s script to students. This script includes a welcome
statement, an overview of WBTSIS, a brief explanation of the evaluation process and the task instruction. The students were requested to familiarise themselves with the system to be evaluated and the evaluation procedures. After student had completed the tasks, the questionnaire was given to them for filling in. The questionnaire is divided into five main sections. The first section contains questions about personal data of the surveyed user, such as his/her level of education and field of work; the second section includes validating teaching framework; the third section tests the ease of use of the tool’s interface; the fourth section tests the tool’s usability; the fifth section includes other general usability aspects; and the final section gives the surveyed user the chance to mention any problems that he/she experienced while using the tool, and to provide suggestions for future improvement.

6.3.4. Evaluation of Results

Based on the questionnaire presented in Appendix B, the result of the survey are divided and presented in the following section.

6.3.4.1 Ease of use

From the data presented in Table 6.1 and the graph presented in Figure 6.1, the strong points for WBTSIS comes from 80 percent of participants admitting that it was relatively easy to move from one part of a task to another, as shown in the eighth column of the graph. Another strong point is the fact that 80 percent of the participants agreed not experiencing any unexpected stoppage by WBTSIS. However, 50 percent admitted that working with it was satisfying.
On the negative feedback on WBTSIS, three participants reported that the software responded too slowly to input while four participants out of the 15 participants were undecided on the speed of the tool.

### Table 6.20: Summary of WBTSIS’s Ease of Use

<table>
<thead>
<tr>
<th>Statement</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>This software responds too slowly to inputs.</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>The software has at some time stopped unexpectedly.</td>
<td>2</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Working with this software is satisfying.</td>
<td>8</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>The software hasn’t always done what I was expecting.</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>It’s relatively easy to move from one part of task to another.</td>
<td>12</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>This software occasionally behaves in a way which can’t be understood.</td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>
6.3.4.2 Teaching Framework Validation

The structure of our proposed teaching framework involves several key ingredients: (i) general guidelines that include software inspection concepts and its techniques (ii) specific guidelines that consist of software inspection process and applying techniques; and (iii) sample activities that contain three component; conceptual samples, work sample and collaboration. It is believed that the structure and composition of the proposed framework is well-suited to overcome the teaching challenges associated with this subject. In order to make an initial assessment of the framework, the following questionnaire was given to 15 students from the faculty of Computer Science.
Table 6.21: Summary of WBTSIS Teaching Framework Validation

<table>
<thead>
<tr>
<th>Q1. Did this framework help you understand the inspection process?</th>
<th>Strongly Agree</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q2. Did this framework simulate learning environment for collaborative work?</th>
<th>Strongly Agree</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q3. Would you find it helpful to solve problem with WBTSIS tool?</th>
<th>Strongly Agree</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 6.21 Teaching Framework Validation
6.3.4.3 System Functionality

In this section, the different functionalities of the system are evaluated to determine how well they perform their functions. The results of system functionality are illustrated in Figure 6.3 and table 6.10. As shown in Figure 6.2, functionality of the WBTSIS is evaluated in term of five important functionalities. The questionnaires are rated on a scale of 1 to 5 where 1 denotes the most positive response and 5 the most negative response. The average for each question is computed using the following formula.

\[ \text{AVG} = \frac{1 \cdot x + 2 \cdot x + 2 \cdot x + 4 \cdot x}{\sum x} \]
### Table 6.22: Summary of WBTSIS’s Functionality

<table>
<thead>
<tr>
<th>Q</th>
<th>Very good</th>
<th>Good</th>
<th>Average</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1. Sign up</td>
<td>1</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Q2. Creating/Modifying general guidelines</td>
<td>1</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Q3. Creating/Modifying specific guidelines</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Q4. Creating/Modifying conceptual samples</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Q5. Creating/Modifying work samples</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Q6. Categorising students and collaborating</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

### Figure 6. 22 WBTSIS tool Functionality
6.4 Conclusion

In this chapter, the testing of WBTSIS with various testing techniques is successfully performed. Furthermore, WBTSIS is evaluated on its functionality, ease of use and teaching framework validation from viewpoint of participants. The objective of the evaluation, the methodology used, and participants’ backgrounds are also included in this chapter. The qualitative measurements and its results are presented in this chapter as well.
CHAPTER 7

CONCLUSION

7.1 Goals and Objectives

This research discussed on developing A Software Support Tool for Teaching Software Inspection aimed to facilitate the teaching of the process of software inspection and its techniques with using blended learning. This thesis was written with the intention to fulfill the three objectives which can be obtained from Section 1.3 in Chapter 1. Below are the outcomes of the achievement for each objective:

4. Objective 1: To develop the teaching framework for software inspection

In teaching software inspection subject over the past years shows that the lack of an appropriate teaching framework to guide student learning such projects resulted in students not gaining the adequate breath of software inspection skill by industry. This motivates us to develop teaching framework that aim to equip students with a well rounded knowledge of software inspection required to success in industry.

This teaching framework was proposed based on blended learning that it explained in more detail in chapter2. Because blended learning, learners and teachers work together to improve the quality of learning and teaching. The aim of blended learning is to provide realistic practical opportunities for learners and teachers to make learning independent, useful, sustainable and ever growing. This teaching framework involves two types of users which they are students and teacher. High scores in student feedback questionnaire indicate that this framework is helpful for student learning through specifying clear objective. This framework
was a success from the perspective of students and fulfills its major purpose of providing students with depth understanding of software inspection skill.

5. Objective 2: To develop the appropriate support tool

This objective has been achieved in which a support tool for teaching software inspection named as WBTSIS; acronym used to denote the first letter of each word of the Web Based Teaching Software Inspection System. It has been developed by using Rational Unified Process (RUP) methodology. WBTSIS supports interaction between teacher and students, as well as access to all guidelines and learning activities besides allowing the users to perform the software inspection process in a group.

The WBTSIS benefits its users by providing guidelines to support students’ learning about software inspection as well as to access the conceptual samples for evaluating students’ level of understanding. It also provides opportunities a collaborative environment where each user is able to perform the software inspection process in a selected group. All the modules and functions provided in WBTSIS are according to proposed teaching framework.

6. Objective: To evaluate the support tool for teaching software inspection

After WBTSIS has been developed, it was tested and evaluated by the target users which are students. The questionnaire (which can be obtained from Appendix B) was distributed to them in order to examine the users’ acceptance, and each of these questions was measured using the four-point Likert scale. The questions were focused on the system’s usability, to find out whether WBTSIS help the users to improve their work and whether or not WBTSIS provide the functions that are useful to the users. The teaching framework is also to examine.
Another important aspect highlighted in the questionnaire was the system’s ease of use, to observe whether WBTSIS is easy to use by both parties (teachers and students). All answers of questionnaire given by the users were analyzed and represented. The results showed that all teachers and most of the students agreed that WBTSIS is a useful teaching tool as well as easy to be used, by giving positive feedbacks. Thus, it can be concluded that WBTSIS has satisfied and fulfilled the perceived usefulness and perceived ease of use aspects.

7.2 Research Contribution

WBTSIS is contributed in software engineering science, especially requirement engineering in following ways:

a) This research reviews existing literatures about software inspection and its techniques.

b) This research investigates the existing learning theories and teaching tool.

c) This research provides a teaching framework for software inspection based on existing teaching framework.

d) This research provides special learning theories for teaching the software inspection process to software engineering student.

e) This research provides way to present a teaching software inspection by using support tool.
7.3 Future Research Works

Further work includes the improvement of the system for better scope and performance. Future research possibilities include using the system for a requirement engineering class to emphasize that collaborative activities do not only support software inspection subject, but any other subjects as well. The teaching framework also will be enhanced. Some other feature for collaboration will be added to WBTSIS tool such as video conferencing and group chatting. By default, WBTSIS will automatically assign students to each group after the students have registered. However, the teacher still can change the number of group members by himself.

This is not a major problem, but still can be improved to prevent any changes.

Since WBTSIS support e-mail to be used internally, by the registered user only, therefore the future work may expand the use of email externally. The e-mail may be send or receive to/from another e-mail provider.
References:


Gintell, J., M. Houde, et al. (2002). Lessons learned by building and using Scrutiny, a collaborative software inspection system, IEEE.


Scenario: Manage Conceptual Samples

6. Teacher login to system
7. Choose conceptual samples
8. Choose examples/tutorial/quiz from main menu
9. Create examples/tutorial/quiz
10. Add examples/tutorial/quiz to database
Scenario: Use Work Sample

11. Student login to system

12. Choose case study

13. Visit case study

14. Choose project

15. Display project

16. View team work and role

17. Choose project and role
APPENDIX B

Evaluation Questionnaire Form for
A Support Tool for Teaching Software Inspection (WBTSIS)

SECTION A: Personally Information

1- What is your age?
   20-25 [ ]  26-30 [ ]  over 30 [ ]

2- What is your major?
   MSE [ ]  MIS [ ]  MCS [ ]  MLIS [ ]

SECTION B: PSTRE Evaluation

Part A: System Ease of Use

(1.Agree, 2.Undecided, 3.Disagree)

1. This software responds too slowly to inputs. 1 2 3
2. The software has at some time stopped unexpectedly. 1 2 3
3. Working with this software is satisfying. 1 2 3
4. The software hasn’t always done what I was expecting. 1 2 3
5. It is relatively easy to move from one part of task to another. 1 2 3
6. This software occasionally behaves in a way which can’t be understood. 1 2 3
**Part B: Teaching Framework Validation**

(1.Agree, 2.Strongly Agree)

1. Did this framework help you understand the inspection process?  
   - 1  
   - 2

2. Did this framework simulate learning environment for collaborative work?  
   - 1  
   - 2

3. Would you find it helpful to solve problem with WBTSIS tool?  
   - 1  
   - 2

**Part C: System Functionality**

(1-Bad, 2-Average 3-Good, 4-Very Good)

1. Sign up  
   - 1  
   - 2  
   - 3  
   - 4

2. Creating/Modifying general guidelines  
   - 1  
   - 2  
   - 3  
   - 4

3. Creating/Modifying specific guidelines  
   - 1  
   - 2  
   - 3  
   - 4

4. Creating/Modifying conceptual samples  
   - 1  
   - 2  
   - 3  
   - 4

5. Creating/Modifying work samples  
   - 1  
   - 2  
   - 3  
   - 4

6. Categorising students and collaborating  
   - 1  
   - 2  
   - 3  
   - 4