CHAPTER ONE

INTRODUCTION

An inspection is generally accepted as a useful technique to find errors in both documents and codes. The technique was originally devised by Michael Fagan over twenty years ago at IBM and had proven to be an effective technique for design, code and test phases (Fagan, 1976). Rigorously applied, inspections have beneficial impact on the productivity and quality of software project development.

The research literature shows that there are several variants of this approach have been proposed for improving inspection performance. These include the Fagan Inspection (Fagan, 1976), Active Design Reviews (Parnas & Weiss, 1985), N-Fold Inspection (Martin & Tsai, 1990) and Phased Inspection (Knight & Meyers, 1991). The variation used depends on the document being inspected, past experience of inspection, team preference and critically of inspection.

The benefits of inspection are direct consequence of its ability to be applied early in the software development lifecycle, to prevent the migration of defects to later phases of software development and improve communication to the entire development project team. The longer the defects remain in the system, the more expensive they are to be removed. This is due to the fact that the cost of removing a defect when the system is operational is up to 1000 times the cost of removal during the requirement stage (Gilb and Graham, 1993). Inspections are cited as one of the nine best practices for software management (Brown, 1996) and it appears at Level 3 of the Capability Maturity Model (Humprey, 1989).

This dissertation focuses on the development of a CASE tool for code inspection. Code inspection is a formal read-through of a program by a group of people with the aim of identifying problems with it and of sharing views on how it could be improved (Kelly et al, 1992). Once an analyst or a programmer has planned and created a program, inspection serves as a way to check for defects and acts to improve the program. According to Fagan, inspection can save 9 to 25 percent of development expense because defects are found before they spread and multiply (Fagan, 1976). Therefore, code inspection should be used to judge the quality of the source codes but not the quality of the people who write the source codes.

1.1 Objectives

The objectives of this research are as follows:

a. To develop a CASE tool for code inspection that can verify certain types of syntax errors written in C programming language.
b. To generate the inspection results in listing format.

1.2 Project Scope

The scope of this research project covers the development of a prototype CASE tool for code inspection. This tool known as *CodeIns*, would ensure that each line of codes written complied with C syntax. The *CodeIns* will generate the inspection outcomes at the end of the code inspection process.

1.3 Overview on Development Strategy

The strategy used in the development of *CodeIns* is based on the software prototyping methodology. This is because it allows all or part of a system to be constructed quickly. This methodology is easy to understand, and can clarify uncertain issues evolve and therefore, would reduce the risks in the development process (Sommerville, 1995).

Prototyping model consists of six steps as shown in Figure 1.1. Like other approaches in software development, prototyping begins with requirements gathering. After identifying the known requirements, a quick design is then formulated. The quick design focuses on the top-level architecture and data design issues rather than on detail procedural design. The quick design leads to the construction of a prototype. The prototype is tested and evaluated to refine requirements. A process of iteration occurs until all requirements are formalized or until prototype has evolved into a production system (Sommerville, 1995).

Prototyping model possess the following benefit: -

a. Requirements are clearly delineated and understood.

b. Important architectural design ideas are validated.

c. Critical user interface issues can be handled.

d. Feedback is available quickly and early in the project.

e. Systems developed can address users' needs and expectations more closely.

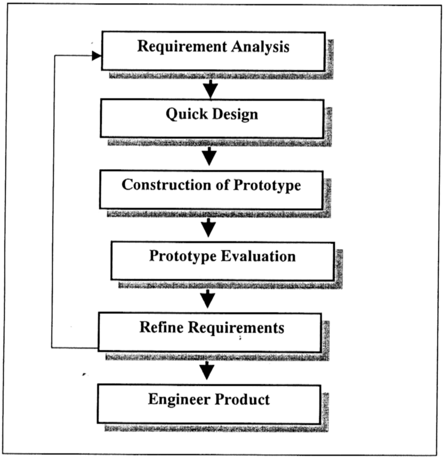

Figure 1.1 Prototyping Model

1. 4 Project Schedule

A systematic project schedule was planned to manage the time and tasks needed to accomplish so that the project objectives can be achieved. Table 1.1 shows the project schedule for *CodeIns* development.

Table 1.1: Project Schedule

Key Activity	July 01	Aug 01	Sept 01	Oct 01	Nov 01	Dec 01	Jan 02	Feb 02	March 02	Apr 02	May 02
Research Proposal	■	■	■								
Literature Review		■	■	■	■						
Requirement Analysis	■	■	■								
System Design			■	■	■						
Coding/ Prototyping				■	■	■	■	■	■	■	■
Testing									■	■	■
Documentation				■	■	■	■	■	■	■	■

1.5 Report Overview

The purpose of this report is to document the essential data gathered and activities performed throughout the development of the project. It covers the project studies and analysis, the design of the software, development and testing stage of the system. User manual of *CodeIns* is included in the Appendix A of this report.

This report is divided into five chapters, which are described as follows:

Chapter 1: Introduction

This chapter gives an overview of the project, the project objectives and scope, and development strategy used for this project.

Chapter 2: Literature Review

This chapter presents an overview of seven of the most common inspection processes and introduces the basic concept of inspection. Comparisons of a number of tools currently available to support code inspection are also mentioned in this chapter. The advantages and disadvantages in the existing tools were identified and led to the main areas of this research.

Chapter 3: Analysis and Design

This chapter includes the system analysis and design planning for the development of *CodeIns.*

Chapter 4: System Implementation and Testing

This chapter describes the development environment and tools used. The system development and testing involved are also explained in detail.

Chapter 5: Conclusion and Recommendation

This chapter summarizes the contents of the dissertation and the contributions of the development of *CodeIns*. The numerous problems encountered and the solutions taken during the project are highlighted here. Finally, it also discusses the strength, limitations and further enhancements for *CodeIns*.