CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

Being software engineers or software developers it is our responsibility to improve the methods used to construct software applications. Solid development methodologies transcend fads and provide a good foundation of the construction of a software-based application. The following techniques were used to define the system requirements of *CodeIns*:

a. **Internet surfing.** As Internet is the largest information warehouse in the world, it is used to get information on client/server architecture, web technologies and existing code inspection.
b. **Literature Review.** Literature review was done by collecting printed documents; handbooks and journal related to the software inspection processes.

3.1 System Architecture

Based on the notation of implementing a web-based client/server application system, a thorough analysis has been carried out to find the most suitable architecture for this system. The client/server architecture was chosen to be the best choice for this system implementation.

Client/server computing involves splitting an application into tasks and putting each task on the platform where it can be handled most efficiently. This usually resulting in putting the processing for presentation to the user or client machine and the data management and storage on the server. Depending on the application and software used, all data processing may occur on the client or is split between the client and the server. The server is connected to the client via a network. Server software accepts data from client and then returns with results to the client. The client manipulates the data and presents the results to the user (Dewiwe, 1993).

The advantages of this architecture are team members need not be aware of where the data resides, how the physical data looks like, and provide the formatting and displaying of the information.

Using web-based client/server architecture, the individual member can do their respective inspection synchronously where the documents can be shared with everyone involved in the inspection process.

3.2 The Programming Technologies and Languages

Once the architecture has been determined, the next step is to find a suitable programming technology and programming language to implement the system. Several key issues need to be addressed when implementing the system such as flexibility, scalability, extensibility, availability, portability and accountability. The chosen programming technology and language need to fulfil the above-mentioned requirements.

3.2.1 Scripting Language

Scripting language can be divide into two types: client-side scripts most often add improved user interface and data validation and server-side scripts are practically unlimited but they are primarily used to capture business rules and access to data. The following scripting languages were used to implement the system requirements of *CodeIns*.

3.2.1.1 JavaScript

JavaScript is a client-side scripting language used for writing small programs that are embedded inside a page of HTML. The page that loads the JavaScript code is interpreted by the browser. JavaScript was developed by Netscape, and is recognised as standard by Microsoft's Internet Explorer as well (though Microsoft originally developed its own version of JavaScript called JScript) (Corning et al, 1997).

By operating on the client-side, JavaScript speeds up simple interactive behaviour without the need to use a network. This reduces server load and speeds up performance at the client side. In general, the use of an embedded client-side scripting language can create a more event-driven page, reacting instantly to user input such as mouse clicks or text entry at the browser (Corning et al, 1997).

3.2.1.2 VBScript

VBScript is a scripting language from Microsoft, a Microsoft's answer to Netscape's JavaScript. VBScript, which is based on the MS Visual Basic, is like JavaScript, embedded as a small program in a web page that is interpreted and executed by the Web client. The scripter controls the time and nature of the execution and VBScript functions can be called from within a Web document, often executed by mouse click function, buttons, ActiveX controls, or other actions from the user. VBScript can be used to fully control compatible browsers, including all the familiar browser attributes (Siler & Spotts, 1998).

VBScript enables authors to create scripts using a subset of the Microsoft Visual Basic language. It does not include functionality that directly accesses the client machine's operating system or file system, and thus, it is safe for the World Wide Web. VBScript is implemented as a fast, portable interpreter for use in Web browsers and applications that use ActiveX controls, java applets and OLE Automation servers (Siler & Spotts, 1998).

3.2.1.3 ActiveX Document

ActiveX document is an application that runs inside a container, such as Internet Explorer, instead of running as a standalone program. The 'document' portion of the name comes from the analogy of word processing documents or spreadsheets. These files contain data but must be accessed by a program in order to be view or edited.

ActiveX document is a full-scale, conventional document hosted in a web browser. It contains part of user interface, merging the application menus with the container menus. ActiveX documents have two key benefits. First, they lower software ownership costs by eliminating the need for client deployment. Second, they allow the user to maintain a single desktop interface, which, in turn leads to lower user training costs and higher user productivity (Corning et al, 1997).

The primary reason to use ActiveX documents in Visual Basic is to create Internet – enabled applications. Creating ActiveX documents provides a number of advantages over creating Internet applications by other means (Siler & Spotts, 1998). Some of the advantages are:

a. Do not have to learn another programming language to create the documents
b. Design Internet application using the Visual Basic design environment
c. Access to Visual Basic's rich debugging environment for testing the code and fixing any problems that arise

3.2.1.4 Active Server Page

Active Server Page (ASP) can run on Windows NT 4.0 Server with Internet Information Server 3.0 (IIS 3.0), Windows NT 4.0 Workstation with Peer Web Services, and Windows 95 with Personal Web Server. In the NT Server, ASP is running in the same address space as IIS, and IIS is running as a service under Windows NT (Johnson et al, 1998).

Active Server Page is the server side execution environment in IIS 3.0 that enables developers to run ActiveX Script and ActiveX Server Components on the server. By combining scripts and components, organizations can create dynamic content and powerful web-based applications easily (Johnson et al, 1998).

ASP technology allows HTML pages to contain complex scripts that are executed at the host, allowing for dynamic content pages to be created from data retrieved from databases or any other resources available to the server. ASP also allows for state management, which lets keep track of users as they navigate through the pages (Johnson et al, 1998).

The Active Server Page is implemented as an Internet Server Application Programming Interface (ISAPI) filter running under IIS. Whenever a Web client makes an HTTP request of a Web server, the Active Server ISAPI filter gets a chance to intercept the request. If the request is for an .asp file, the Active Server takes over from IIS, parses the entire file from top to bottom, processes the server scripts(s), and returns an HTML output file to IIS. IIS then returns this data stream to the requesting Web client (Corning et al, 1997).

3.3 Requirements Analysis

Requirements analysis covers the area of functional requirements and non-functional requirements of *CodeIns*. A complete understanding of software requirements is essential to the success of a software development effort.

The main function of *CodeIns* is to inspect lines of source codes. The system is divided into two modules that are explained in the system functional requirements.

3.3.1 Functional Requirements – System Security

An authentication and authorization process is vital to *CodeIns* to protect its use from unauthorized users. Therefore, a user identity and a password are required to access the *CodeIns* modules.

3.3.2 Functional Requirements – File Module

The file module is divided into two sub-modules namely Open File and Close File.

3.3.2.1 Open File Sub-Module

This module enables an inspector to choose any C language files. The coding would appear in another frame. Hence, the inspector can go to the inspection module to perform the inspection process. If the source codes contain syntax error, the inspector can edit the source code file and then save as the same file name.

3.3.2.2 Close File Sub-Module

This module closes the text area that contains the source codes and links to a blank frame.

b. Functionality Description

The system also provides functionality description for each of the module as a guide for the user while using the system.

c. User Friendliness

In order to achieve ease of use, it is important to have a user-friendly interface. *CodeIns* apply the Graphical User Interface (GUI) approach for better visual effects to the users. The usage of suitable and meaningful captions would help the users to use the system with confidence. An appropriate menu is provided to give the users an overall view of the system. The related functions are also grouped together for ease of selection.

Effective error handling and validation procedures also help the users while using the system. A message would be displayed whenever any error occurs while the users navigate the system.

d. Security

Valid user ID and password are required whenever a user access the system to prevent unauthorized user from using the system.

e. On-line Help

The system also provides an on-line help to guide the user while using the tool. The Help module consists of help contents and context sensitive links to related topics.

3.4 System Requirements

These requirements can be divided into server hardware requirements, server software requirements, client hardware requirements and client software requirements.

3.4.1 Server Hardware Requirements

The server computer requirements are:-

a. A server with at least Pentium 166Mhz processor

b. At least 32 MB RAM

c. Network Interface Card (NIC) and network connection with recommended bandwidth at 10bps or more

d. Other standard computer peripherals

3.4.2 Server Software Requirements

To host and run the system, the server is required to have various supporting software installed as shown in Table 3.1 below:

Table 3.1: Server Software Requirements

Software/Component	**Description**
Windows 95 or 98	Network operating system
Personal Web Server	Web-server service
Microsoft Internet Explorer 5.0	Precondition for VBScript or JavaScript

3.4.3 Client Hardware Requirements

The client hardware requirements only required to have a reasonable amount of RAM and a reasonable quality dial-up connection line. The recommended configurations are:

a. At least 16 Megabytes of RAM

b. Network connection through existing network configuration or modem (recommended at least 14.4kbps)

3.4.4 Client Software Requirements

The client software requirements fall on the browser used by the user. It requires a system that can run Microsoft Explorer 5.0 and above or any other browsers that support ActiveX Document.

3.5 System Design

Design phase is the stage of system development where the requirements for the system are translated into the system characteristics.

3.5.1 *CodeIns* Architecture

Ordinary client/server based application is developed using two-tier client/server architecture. The client workstation performs the displaying and formatting of information to user. Thus it will involve a lot of processing, which with results as a fat-client. On the other hand, the server would routinely provide access to the data. Hence, the development tasks in *CodeIns* would be easier and flexible using this type of architecture.

Focusing on web-based two-tier client/server architecture, the *CodeIns* services can be divided into two types of services which are user services and business services. Both services are explained as follows:-

User Services. At the user services level, authorized user can use a browser to input data. HTML forms are used for authorized user to input information.

Business Services. Merely a processing engine at this level which resides on the machine running Personal Web Server. Request and responses are controlled by written codes specifying its business rules. Figure 3.1 shows the overview of *CodeIns* web-based architecture.

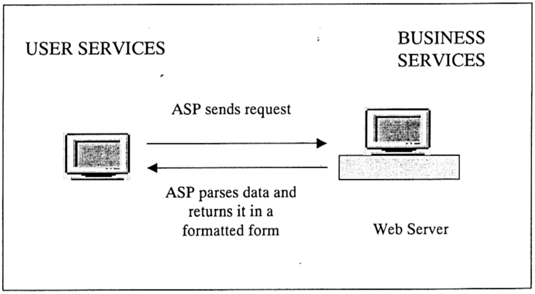

Figure 3.1 *CodeIns* web-based architecture

3.5.2 User Interface Design

The user interface design is based on the GUI approach. Some of the human computer interface (HCI) general principles of designing an interactive system have been considered and applied. These HCI principles among others are consistency, recoverability, confirmation and verification message, responsiveness and reverse action (Alan et al, 1998).

As information in *CodeIns* is displayed in web pages form, it is also important to consider the web pages design. Listed below are some of the considerations taken while designing the user interface of web pages (Alan et al, 1998). They are:

a. Page layout and presentation, that is, does the page looks like it is supposed to when rendered by the browser?
b. Does the page appear as it is supposed to when rendered by different browsers? This is important when using non-standard HTML tags (extension) where different browsers may handle these non-standard tags differently.
c. Are the page elements (graphics, animations, font sizes, and so on) of acceptable size when viewed at 640-by-480 resolutions?

3.5.2.1 *CodeIns* Screen Design

CodeIns screen design is presented in the form of web document on the browser. User's needs, skills level and preferences are of major considerations here. Document should be displayed in an organized pattern. *CodeIns* is a web-based tool for detecting errors. It is important to make sure that the screen design is user-friendly and easy to use.

3.5.2.2 Welcome Page Design

A welcome page is first displayed before going further into the system. Brief introductions on some of the basic functions are displayed.

3.5.2.3 Login Design

This page allows user login using valid logon user identity (ID) and a valid password.

3.5.2.4 Module Design

Overall, there are three modules in this system.

a. File
b. Inspection
c. Help

a. Open File

An inspector can choose any C language file from the server or directory and the document will appear on the screen. The inspector can modify the source codes and then save the file using the same file name.

b. Inspection

An inspector can use *CodeIns* to inspect the source document to detect syntax errors hidden in the source codes. The errors will appear once the syntax error menu is clicked.

c. Help

Instructions of how to use *CodeIns* are provided under this module.

3.5.3 *CodeIns* Workflow

Workflow systems constitute some of the most powerful environments that enable collaborative computations to automate work processes. In developing *CodeIns,* synchronous activities were used to prepare the meeting for detecting errors of the source codes. Figure 3.2 shows schematically what the user sees during the synchronous part of the *CodeIns*. Workflow does not have to be structured. Its primary function is to facilitate the fulfillment of projects and deliverables by a team. Workflow can be defined as the *activities* (work) and *interaction* (flow) required to carry out a business process (Hoffer et al, 1999).

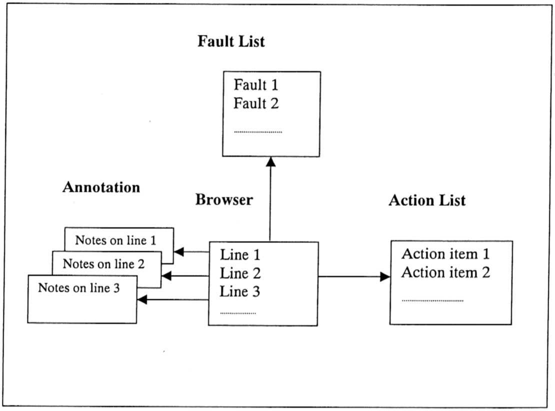

Figure 3.2: Synchronous activities of *CodeIns*

The details of workflow discuss the structured charts and process flow of *CodeIns.*

3.5.3.1 Structure Chart

CodeIns is depicted using the structure chart as shown in Figure 3.3. The use of structure chart is to describe the interaction between modules. *CodeIns* is divided into three major components namely, File Module, Inspection Module and Help Module.

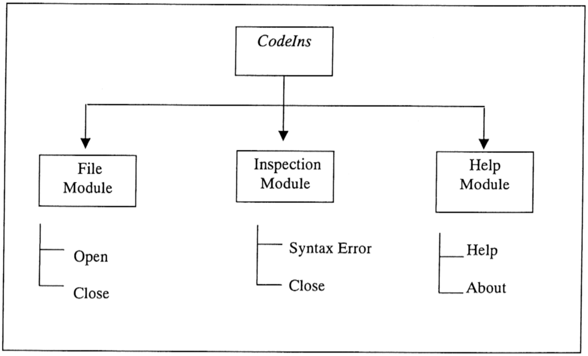

Figure 3.3: Structure chart for *CodeIns*

3.5.3.2 Process Flow

The process flow starts with the explanation about the general flow of code inspection. Represented in Figure 3.4, basically, the flow chart only illustrates how the system handles user login and displays the user's menu.

Once the user has login to the system, it will verify the user. The system will display the respective main menu if the user login is successful.

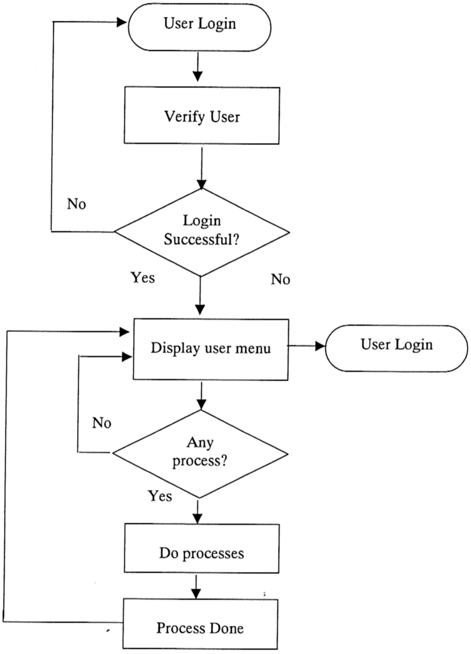

Figure 3.4: General flow of *CodeIns*

The process flow is represented in Figure 3.5. An inspector will get the source codes and then performs inspection phase by phase. The process should start with browsing the file and then inspect the source codes. If the source codes contain errors, the inspector can edit the source codes and then save the file using the same file name.

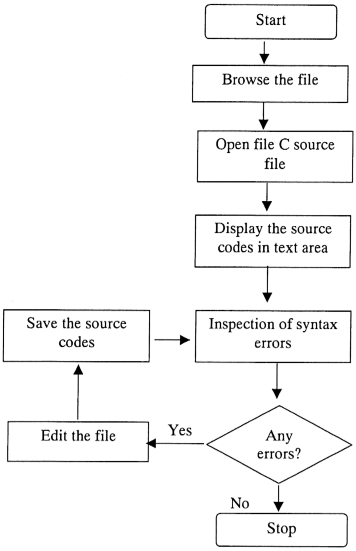

Figure 3.5: Process flow of *CodeIns*