

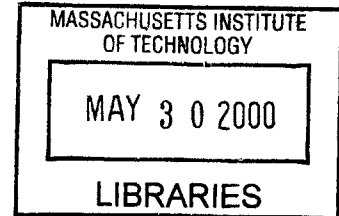
ENG

COLLABORATIVE GRAPHICAL USER INTERFACE DESIGN

By

Hermawan Kamili

B.S. Civil & Environmental Engineering
University of Washington, 1999



Submitted to the Department of Civil & Environmental Engineering
In Partial Fulfillment of the Requirement for the Degree of

Master of Engineering
In Civil & Environmental Engineering at the
Massachusetts Institute of Technology

June 2000

© 2000 Hermawan Kamili All Rights Reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly
paper and electronic copies of this thesis for the purpose of other works

Signature of Author _____
Department of Civil & Environmental Engineering
May 5, 2000

Certified by _____
Professor Feniosky Peña-Mora
Associate Professor, Civil and Environmental Engineering
Thesis Supervisor

Accepted by _____
Professor Daniele Veneziano
Chairman, Department Committee on Graduate Studies

COLLABORATIVE GRAPHICAL USER INTERFACE DESIGN

By

Hermawan Kamili

Submitted to the Department of Civil & Environmental Engineering
In Partial Fulfillment of the Requirement for the Degree of

Master of Engineering
In Civil & Environmental Engineering
Massachusetts Institute of Technology

Abstract

Graphical user interfaces are means of interaction between the user and the computer system. It is important to create a graphical user interface that serves to effectively increase user productivity.

Collaborative Graphical User Interface Design (CGUID) is a method that combines the incremental software development model and team collaboration process. During the incremental software development process, each phase generates products that serve as inputs for subsequent phases. Teams in each phase can collaborate to exchange information, to eliminate problems, and to enhance the products' quality.

CGUID method introduces team collaboration in a continuously development phase so that value is constantly added until the final goal, a high usable graphical user interfaces that increase productivity, is created.

Thesis Supervisor: Feniosky Peña-Mora

Title: Associate Professor of Civil and Environmental Engineering

Acknowledgement

I would like to thank the following individuals for their contributions to my learning at Massachusetts Institute of Technology.

To my parents for their support throughout my educational career.

To Professor Feniosky Peña-Mora for his time and effort in reviewing and commenting this thesis.

To Alan Ng, Steve Kyauk, Teresa Liu, Nhi Tan, Erik Abbot, Wassim El-Solh, Paul Wong, Saeyoon Kim, Kenward Ma, Ivan Limansky, and Kaissar Gemayel for making the Information Technology track was a success.

Finally to all Masters of Engineering students, the Class of 2000, for making me believe that I can achieved all my goals at MIT and beyond.

Table of Content

CHAPTER 1	8
1.1 DESCRIPTION.....	8
1.2 METHODOLOGY	9
1.3 OUTLINE.....	10
CHAPTER 2	11
2.1 INTRODUCTION	11
2.2 BUSINESS ADVANTAGES OF GUI	12
2.3 GRAPHICAL USER INTERFACE	15
2.4 SUMMARY	21
CHAPTER 3	22
3.1 INTRODUCTION	22
3.2 SOFTWARE DEVELOPMENT MODEL.....	27
3.3 USER CENTERED DESIGN	30
3.4 SUMMARY	31
CHAPTER 4	33
4.1 INTRODUCTION	33
4.2 COLLABORATION IN THE ANALYSIS PHASE	34
4.3 REQUIREMENT ANALYSIS.....	37
4.4 TASK MODELING.....	40
4.5 SUMMARY	44
CHAPTER 5	46
5.1 INTRODUCTION	46
5.2 COLLABORATION IN THE DESIGN PHASE.....	47
5.3 USER OBJECT MODELING	49
5.4 STYLE MODELING.....	52
5.5 GUI DESIGN	55
5.6 SUMMARY	58
CHAPTER 6	60
6.1 INTRODUCTION	60
6.2 COLLABORATION IN THE CONSTRUCTION PHASE.....	61
6.3 GUI PROTOTYPING	62
6.4 GUI TESTING	63
6.5 SUMMARY	65

CHAPTER 7	67
7.1 INTRODUCTION: MEETING MANAGEMENT TOOL	67
7.2 IECOLLAB REQUIREMENT ANALYSIS – STUDY CASE.....	68
7.3 IECOLLAB GUI DESIGN.....	74
7.4 IECOLLAB GUI DEVELOPMENT.....	80
7.5 SUMMARY	85
 CHAPTER 8.....	 86
8.1 INTRODUCTION: COLLABORATIVE GUI	86
8.2 PROBLEMS DATA COLLABORATION.....	87
8.3 TECHNOLOGY FOR COLLABORATIVE GUI	89
8.4 SUMMARY	92
 CHAPTER 9.....	 94
Ch 9 / pgs.94 - 95 have bben omitted.	
 REFERENCES.....	 96

List of Figures

Figure 2-1	GUI vs CUI in Proficiency.....	13
Figure 2-2	Return of investment on GUI	14
Figure 2-3	GUI's Elements	16
Figure 2-4	GUI as an Application	17
Figure 2-5	Webpage.....	18
Figure 2-6	GUI with Java	20
Figure 2-7	GUI with JavaScript.....	21
Figure 3-1	Collaboration Graphical User Interface Design Concept.....	22
Figure 3-2	Collaboration Graphical User Interface Design.....	22
Figure 4-1	Requirement Analysis	33
Figure 4-2	Task Modeling.....	34
Figure 4-3	Arousal Performance Relationship	40
Figure 4-4	Task Model	42
Figure 4-5	Task Table.....	43
Figure 4-6	Task Scenario	44
Figure 5-1	Object Modeling	46
Figure 5-2	Style Modeling	47
Figure 5-3	GUI Design	47
Figure 5-4	Example of object, attributes, and action.....	51
Figure 5-5	Relationship Table.....	52
Figure 5-6	Window Hierarchy	55
Figure 6-1	GUI Prototyping	60
Figure 6-2	GUI Testing	61
Figure 7-1	Requirement Analysis Recommendation for the Main Window.....	75
Figure 7-2	New Designed Main Window.....	76
Figure 7-3	Schedule Meeting Window Design.....	77
Figure 7-4	ieCollab Main Window	83
Figure 7-5	ieCollab Create Meeting Window	84
Figure 7-6	ieCollab Edit Meeting Window	84
Figure 8-1	Web Wrapper Technology.....	90
Figure 8-2	Context Mediation Engine	91
Figure 8-3	Excel Interface as Collaborative Display.....	93

List of Tables

Table 3-1	CGUID Components	23
Table 3-2	Products of Collaborative Graphical User Interface Design	24
Table 3-3	Collaborating Teams.....	28
Table 4-1	Analysis Team.....	35
Table 5-1	Design Team	48
Table 5-2	Dynamic Modeling.....	52
Table 5-3	8 Golden Rules in Design Principles by Shneiderman	52
Table 6-1	Construction Team.....	61
Table 7-1	ieCollab User Profile	69
Table 7-2	User Case for Schedule Meeting	73
Table 7-3	Use Case for Edit Meeting.....	73

Chapter 1

Introduction

1.1 Description

User Interfaces (UI) allow human users to interact with computer systems. They are the media to communicate with and perform tasks on the computer systems. Without user interfaces, users have no access to the interior of the computer systems.

Typical user interfaces in the routine information systems include the computer hardware such as keyboards, mouses, computer monitors, and on/off switch. Another type of for user interface is the image display on computer monitors such as windows, menus, and message dialogs. This image display, then, is known as Graphical User Interface (GUI).

Research in Xerox Palo Alto evaluated GUIs which replaced the character display and command line of the 1980's, with large bit-mapped display, icon, multiple windows, and pointing devices called a mouse [Peddie, 1992]. The result of research showed that people could learn to use application with a graphical interface more quickly than with command lines. The graphical interfaces were also easier to remember, and helped users become more productive.

As times continued, graphical user interfaces did not just appear in the form of software application. In the beginning of 1990's, a new technology, called the Internet, enabled new forms of graphical user interfaces through Web pages.

Utilized through as applications and Web pages, the graphical user interfaces play an important role in adding value to businesses, while increasing productivity.

This thesis shows how to design graphical user interfaces, which serve as tools whose purpose to effectively increase productivity.

1.2 Methodology

Collaborative Graphical User Interface Design (CGUID) arises from the author's experience in the ieCollab project. The author also uses fundamentals of Graphical User Interfaces Design and Evaluation (GUIDE) process invented by David Redmond-Pyle and Alan Moore.

In the ieCollab project, a project held by the Information Technology program in the Department of Civil & Environmental Engineering at MIT, the author learned about the collaboration process with distributed teams in software development.

The GUIDE process, on the other hand, introduces the connectivity of several lower level design stages and the interaction of each stage with other stages. The lower level design stages consist of Analysis, Task Modeling, Style Modeling, Object Modeling, GUI Designing, GUI Prototyping, and GUI Testing. The notion of connectivity arises when each design stages create outputs that will be used as inputs for other design stages. For example the analysis stages will create user task that will become an input for Object Modeling, other lower level design stages.

Each lower-level design stage is associated with high-level design stages. Requirement Analysis and Task Modeling are associated with Analysis Phase. Object Modeling, Style Modeling, and GUI Design are associated with Design Phase. Finally, GUI Prototyping and GUI Testing are associated with Construction Phase.

CGUID combines the learning of collaboration process in ieCollab project and graphical user interfaces development using GUIDE process. It explains how collaboration between teams can be implemented in the GUIDE to create a graphical user interface for increasing productivity.

1.3 Outline

Chapter 2 is an overview about the types of graphical user interface. Chapter 3 is the guideline for graphical user interface design. In Chapter 4 through 6 describes the higher level design methods of analysis, design and construction.

The following two chapters (Chapter 7 & 8) are about the case study of ieCollab GUI development process and future development in graphical user interface.

Chapter 9 concludes the thesis.

Chapter 2

Overview of Graphical User Interface

2.1 Introduction

Graphical User Interface was envisioned by Van-nevar Bush in 1945 [Peddie, 1992]. The development of GUI continued when Ivan Sutherland designed Sketchpad in the early 1960s for his graduate thesis [Peddie, 1992].

From the 1950s to the 1970s, scientists at the Stanford Research Institute (SRI) and Xerox Palo Alto Research Center (PARC) investigated the interactive user interface [Peddie, 1992]. The results showed that people learned to use applications with a GUI more quickly than learning command lines. GUI increase productivity, plain, and simple. GUI reduce the time a user spent resolving interface issue, which allows the user to perform tasks more efficiently.

It was not until 1984, when Apple introduced the Macintosh, that the window environment reached the average consumer. The Macintosh GUI has changed the users' expectations of a computer interface from being complicated to more friendly tools [Mayhew, 1984].

2.2 Business Advantages of GUI

The main purpose of a graphical user interface is to make the computer more functional for the user. A common graphical user interface is another way of reaching the same goal with several additional benefits. A GUI environment offers familiar symbols or icons that resemble commonly known functions or items such as files and trash cans so that complicated, arcane command line sequence do not have to be memorized [Redmond-Pyle, 1995].

For some pedantic groups, the benefit of using GUI is a debatable issue. However the argument is more like to be preferred issue. For those power users who need the “feel of the machine” and do not mind wasting time trying to figure out the subdirectory structure path, configuration system, having the arcane command line sequences is not a big issue. GUI or non-GUI let the - “may the operating system be with you”. However for the rest of novice mere mortals who just want to get to an application, do a little work and go home, GUI is a godsend [Peddie, 1992].

The business climate of 1990s is extremely competitive, and most organizations are seeking to improve their business performance, by implementing strategic programs of Total Quality Management of Business Process Re-engineering. For many organizations, improving the quality of customer service and increasing the productivity of office staff are among the critical business objectives. Software with high usability can make an important contribution to meeting these business needs.

In addition to the quantity of work, there is the quality of work of the user. A well designed GUI can reduce the scope for user errors. Fewer errors provide higher quality of service, and may lead to an increase in productivity, as less time is spent on reconciliation or sorting our consequences.

2.2.1 Learning Curve

High learnability reduces training time and cost, which is often a significant proportion of the installation costs of a new system. In addition to this immediate benefit, it can have a wider commercial significance. Having a system which is easy to learn and easy to use can enable

more flexible staffing practices, as staff moved from one system to another become effective more quickly. More radically, it may be possible for a single member of staff to use several systems, moving between them as required to perform all aspects of a complex task, where previously the work was fragmented between several people using different system. In this way the learnability of a GUI can be critical to the success of a business process re-engineering initiative [Peddie, 1992].

According to survey, a common quote from users and managers on GUIs versus text based system is that a typical GUI-based user works with six applications, while a character-based user (with a Character User Interface or CUI) work with no more than three or four application. This is because the more complex learning curve associated with the CUI. Users gain greater proficiency more quickly with a GUI [Peddie, 1992]. Figure 2-1 describes the difference in proficiency between GUI and CUI users.

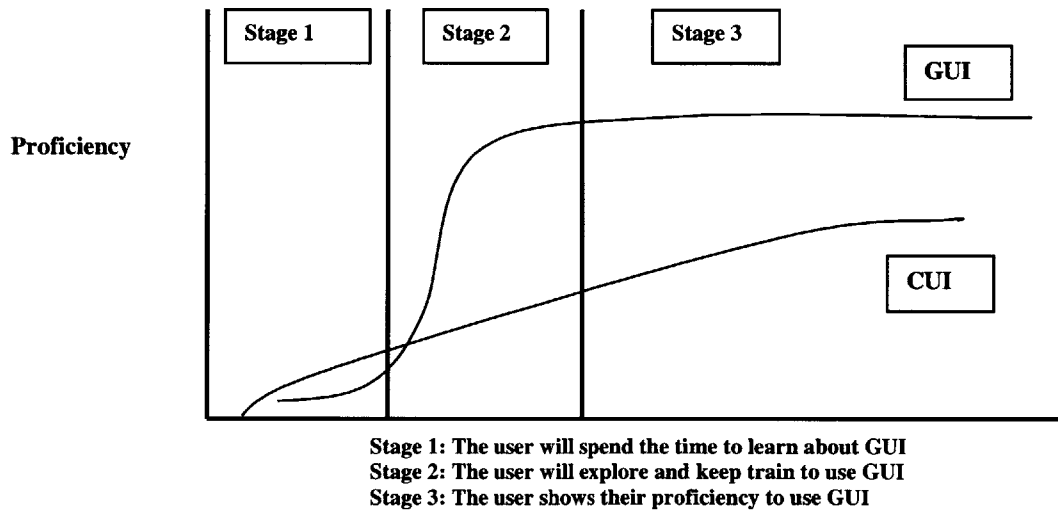


Figure 2-1 GUI vs CUI in Proficiency

Source: Peddie, Graphical User Interface and Graphics Standards, 1992

The flexibility of a user interface is also important for business. Most organizations have an on going change in their structure, products and operating procedures, and need software systems

to be flexible to support users changing their business practice. Lack of flexibility either means a barrier to changes, or expenditure on regular amendments to the user interface to maintain competition.

2.2.2 Return on Investment of GUI

A GUI will enhance the productivity of the individual. Equipping a user with a system that has a GUI is inexpensive relative to an organization's overall cost. As the user gets familiar with the GUI, the productivity will increase rapidly and but eventually taper off. This is known as the asymptote of productivity. However the organization can see a rapid but limited gain in productivity or return on investment (ROI). Figure 2-2 describes the return of investment of categories in a corporation related to GUI.

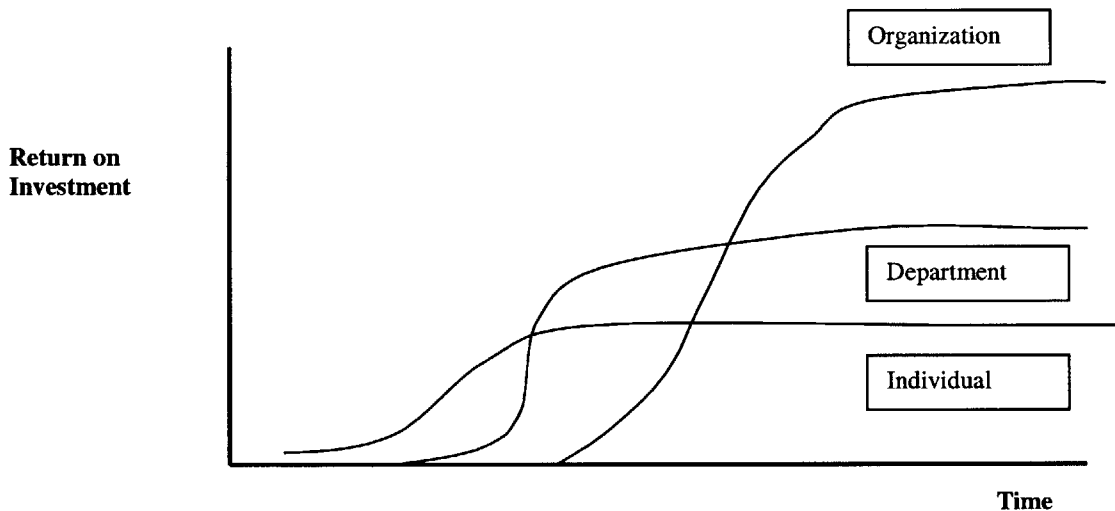


Figure 2-2 Return of investment on GUI

Source: Peddie, Graphical User Interface and Graphics Standards, 1992

If an entire department is equipped with GUI and true workgroup activities are employed, there will be a greater gain in productivity or ROI. However, the gain does not come as quickly. It takes longer to get the department working smoothly together. Hardware and software costs are obviously greater for a department than for an individual. However, the ROI per employees

is greater than for just the single employee. Also, the ROI curve continues to go up over a longer period of time and it has an asymptote.

When an organization invest in an enterprise-wide implementation of GUIs, it takes even more time and money to see any return. However, the return is almost twice as much per employee, and it continues to increase for a much longer time [Peddie, 1992].

So far, the author has described the advantages of using GUI as easily to use tools. In the next section, the author will describes what types of GUI that are currently available in the context of human interaction tools with computer system, as software applications and web pages.

2.3 Graphical User Interface

Most graphical user interfaces come in the form of software application at its earlier time in 1984 [Peddie, 1992]. Furthermore in the early 90's, the Internet technology was opening a new way of communication [Stein, 1997]. The presentation of graphical user interfaces are not only related to the personal computer applications but also a new type of graphical user interface form that allows a user to communicate to computer system (servers) through Internet. They are called web pages.

In the next section, the author will describes the graphical user interfaces as applications and web pages.

2.3.1 GUI as software application

As a software application, the graphical user interfaces mainly consist of the following elements:

- Windows, which graphically display what the computer is doing.
- On screen menus, that can appear or disappear under pointing-device control.
- Icons, which represent files, directories, application, and utilities.
- Dialog boxes, button, slider, checkboxes, and several other graphical widgets that let the user instructs the computer about what should be done.

- Object action paradigm. This is also known as the modeless interactivity where the user indicates the object first and then gives the command.

The user can perform their tasks directly on the screen and use the metaphors as help tools. The metaphor is a figure of speech that says the object is created by implicit comparison or analogy. Using metaphors, GUI can be described as an analogy of a person's work. For example analogy of a desk, GUI can use metaphors to represent the idea of desk. The result is called desktop metaphors like described in Figure 2-3.

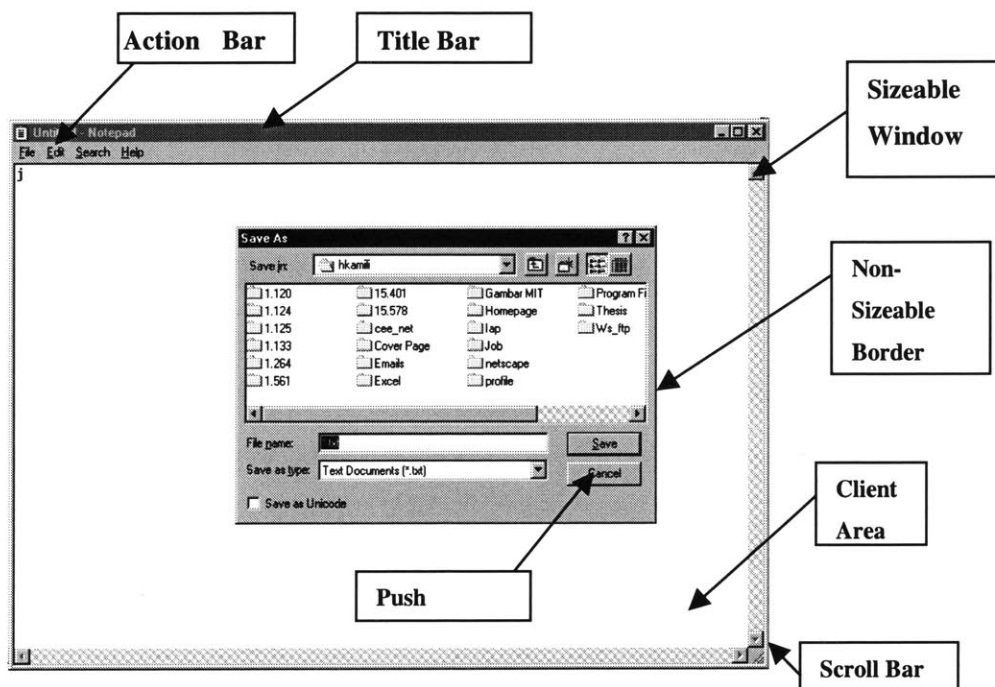


Figure 2-3 GUI's Elements

The icons or buttons, as part of the metaphors, briefly describe function or object that can be executed in the applications. The icons and buttons reduce screen space compared to the corresponding textual description of the functions. Also, they can be understood more quickly if it is well designed. Figure 2-4 describes the use of icons as metaphors in GUI as an application.

Overall, the GUI application replicates objects and functions to complete task and portrays them as on different medium, a screen. The user is able to manipulate these objects and functions by accessing and modifying them. Unlike the command line, the graphical user interface allows the

user to concentrate more on the data instead of the application and tools. The physical organization of the application is hidden and not a distraction [Galitz, 1994].

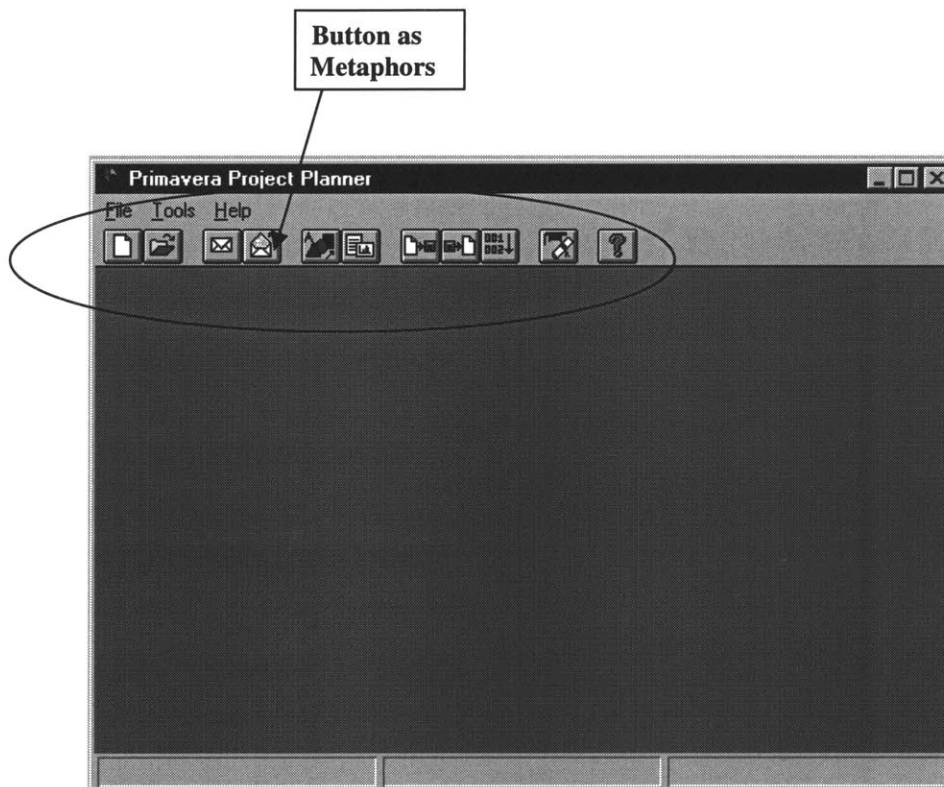


Figure 2-4 GUI as an Application

2.3.2 Webpages

The web is represented as numbers of pages (webpages) which can be view with a browser, an application to retrieve information via the various Internet protocols.

As mentioned earlier, a graphical user interface is the means of communication between a human and a computer, the pages communicate a user to a server that support the pages he/she sees. The servers can be in the user's computers, or anywhere else. Therefore technically, a webpage is a graphical user interface. The figure 2-5 shows examples of pages that allow user to communicate the servers through the Internet.

The pages are written in code called High Markup Language (HTML). HTML specifies the structure of a document using logical terms such as "level 1 header," "ordered list," and "emphasized text". It does not dictate the appearance of the document such as what typefaces to use or how the line breaks on the page. It is the Web browser's function to handle the actual page layout based on the hardware platform capabilities. The user sees the page after the HTTP is translated into pages by the user's browser. The web page in Figure 2-3 is not the same as an application in the Figures 2-3 and 2-4. The pages only connect the user to other page according to the user request. While the application can perform other tasks, in addition to connecting web pages to web pages.

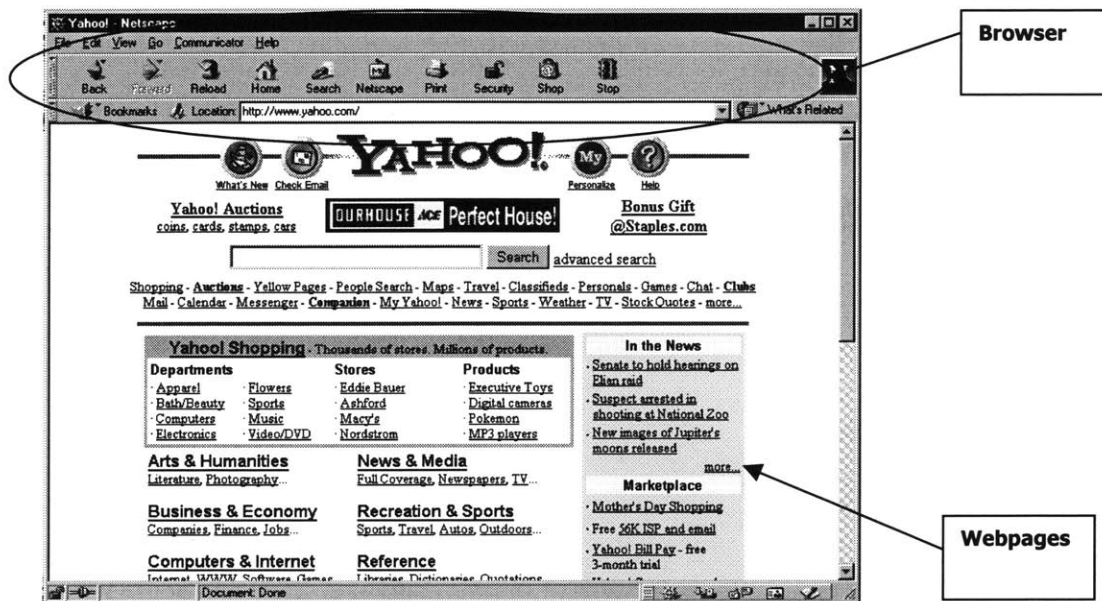


Figure 2-5 Webpage

The next section the author will describe the tools to technology current technology that mostly used for creating application and web pages. The prototyping tools for developing an application or composing a webpages are large in number. They vary from their capability and characteristics in developing the graphical user interfaces.

2.3.3 Java and JavaScript

To have an application being accessed through the Internet, a programming language called Java was invented by Sun MicroSystem. Java is a full-featured programming language that is intended for general use as well as for Web programming. Because its coincidence of being invented in the time where Internet is highly used, Java become the language of choice for implementing cross-platform applications for distribution on the Internet [Stein, 1997].

The widely used browser, Netscape and Microsoft Explorer, have the ability to download and execute small Java programs called Java applets. These applet are imbedded inside the HTML. As an application prototyping tool, Java programs create graphical user interface displays and functions such as open their own windows, create menu bars, create animations and pictures on the fly, and even opening connection to its own home server. Although Java is relatively new when compared to other programming language such as C++ and C, it has been used to create useful functions such as image map with animation and sound, interactive buttons, wire-frame model viewer, and online mortgage payment calculator [Stein, 1997]. An example of Java programming application is shown in Figure 2-6.

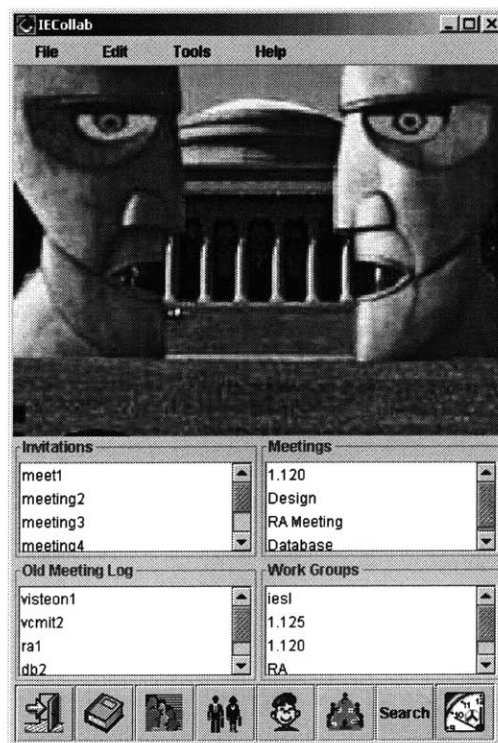


Figure 2-6 GUI with Java

Beside Java programming, there is another language called JavaScripts. JavaScript is a scripting language that was designed specifically for Netscape browsers. JavaScript is embedded directly in HTML pages using several new and extended HTML tags. When a Netscape browser downloads an HTML page can contains JavaScript codes, it begins to execute the script.

JavaScript is the most useful for creating HTML pages with intelligence. For example, one task at which JavaScript really shines validating fill-out forms: a JavaScript program can quickly scan through the content of a fill-out form to make sure that all required fields are presented and have the right format. If something is wrong with the form, the script can either alert the user if he/she did not fill some field in the form incorrectly [Stein, 1997]. Figure 2-7 shows the pages built with JavaScript.

But Java and JavaScript are not the same. Java is a full programming language and made by Sun Microsystems. JavaScript, on the other hand, is a scripting language that was designed specifically for Netscape browser (version 2.0 and up).

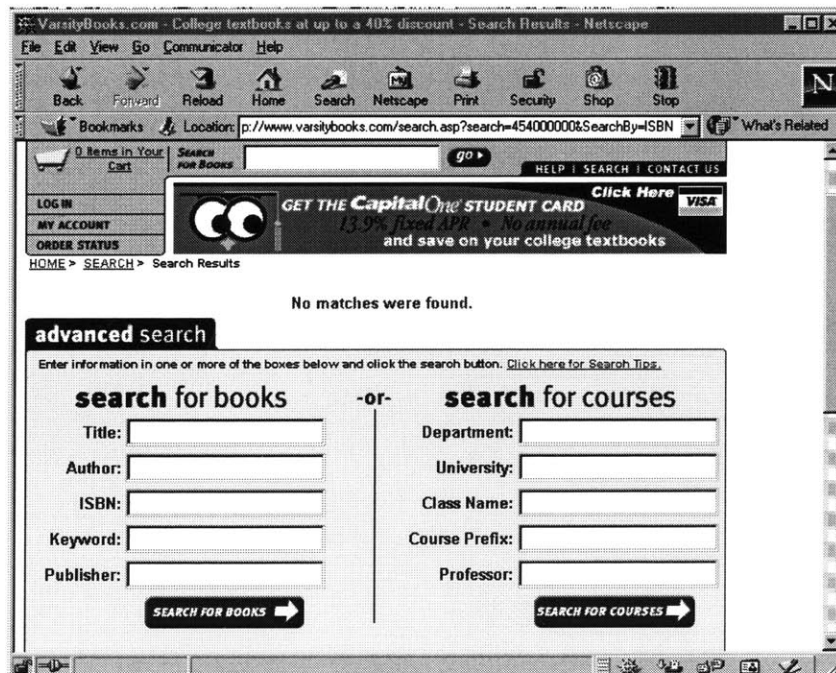


Figure 2-7 GUI with JavaScript

Source:www.varsitybooks.com

These are two prototyping tools that are mostly used by the developer and web composer to create graphical user interfaces as applications and web pages.

2.4 Summary

Graphical User Interfaces came from the idea of simplifying human–computer communication. It has the metaphors to do these and the benefit of having the simpler presentation creates higher outcome in learning curve, return of investment, and profitability.

In previous times when the GUI began to be implemented due to its simplicity, the GUI is competing with the command line user interface. However today, most of the applications are rated due to its value to increase the productivity.

Today with the Internet technology, the graphical user interface does not only used to increase productivity, but profitability from the business point of view. For example, a web site with a good presentation invites the buyer to buy products or services from the web site. The better the presentation, easy interaction, of the graphical user interface, the more web surfer feels like spending the money on the web site. Moreover, if the web site has the non-profit purpose to give information to the user, then it has to be equipped with the good graphical presentation so that the user can find the information easily.

The next chapter introduces the idea of GUI design method, CGUID that implements collaboration in low level design steps.

Chapter 3

Fundamentals of Collaborative Graphical User Interface

3.1 Introduction

Collaborative Graphical User Interface Design (CGUID) is the method of combining the software development process and team collaboration model (Figure 3-1).

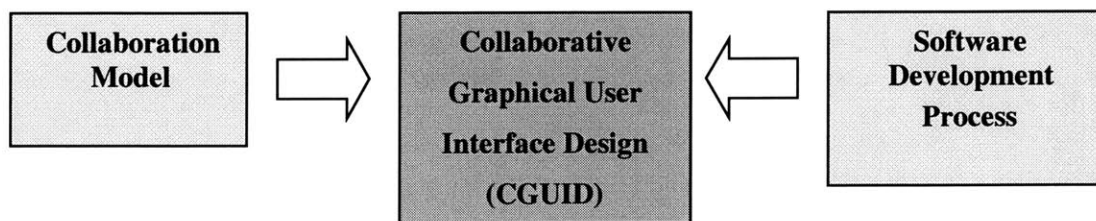


Figure 3-1 Collaboration Graphical User Interface Design Concept

The software development process introduces the idea of gradual phases - analysis, design, and construction - and how each phase adds value to the software development process. In CGUID, each phase adds value by generating products that will be used by the subsequent phases. With the style of transferring products, each phase interconnects with other phases in order to create the final graphical user interface prototype.

Collaboration between teams is the second fundamental in Collaborative Graphical User Interface Design. Collaboration adds value by having each team to contribute their expertises and skills during each of the three phases.

3.1.1 Software Development Process

The development process consists of high-level and low-level design elements, which are expressed in Table 3-1.

Table 3-1 CGUID Components

High Level Design Phases	Low Level Design Stages
Analysis Phase	Requirement Analysis Task Modeling
Design Phase	Object Modeling Style Modeling GUI Design
Construction Phase	GUI Prototyping GUI Testing

Requirement Analysis and Task Modeling are associated with the analysis phase (Chapter 4). Object Modeling, Style Modeling, and GUI Design are associated with the design phase (Chapter 5). Finally, GUI Prototyping and GUI Testing are associated with the construction phase (Chapter 6).

The GUI development method that involve the interconnectivity of several low-level elements was first introduced by David Redmond-Pyle and Alan Moore in their GUIDE (Graphical User Interface Design & Evaluation) model in 1997. This model creates the foundation for the Collaborative Graphical User Interface Design method.

The author chose the GUIDE method as the fundamental of CGUID because of the connectivity between the low-level design elements. The output of each low-level design element becomes

the input for other low-level design elements. For example, the user profile, a product from the Requirement Analysis, is used as input for Object Modeling and Task Modeling. Thus, each low-level element continuously adds values to other low-level elements until they produce the desired graphical user interfaces.

Table 3-2 and Figure 3-2 describe how the interconnectivity works between elements.

Table 3-2 Products of Collaborative Graphical User Interface Design

Product	Element Creating Product	Element Utilizing Product
1. User Profile	Requirement Analysis	Style Modeling Task Modeling Object Modeling
2. Usability Requirement	Requirement Analysis	Style Modeling GUI Testing
3. Task Models & Scenarios	Task Modeling	Object Modeling GUI Design GUI Prototyping GUI Testing
4. User Object	Object Modeling	GUI Design
5. GUI Design	GUI Design	GUI Prototyping GUI Testing
6. GUI Prototype	GUI Prototyping	GUI Testing
7. Evaluation Report	GUI Testing	Requirement Analysis

In addition to interconnectivity of **products** from each low-level design element to other low-level design elements, the CGUID implements the idea of collaboration of **teams** that involves in each high-level design phase.

The teams, represented as analysis, design, and construction teams, collaborate to exchange knowledge and skills for achieving the high-quality products for each phase. The following section explains the idea of collaboration in more details.

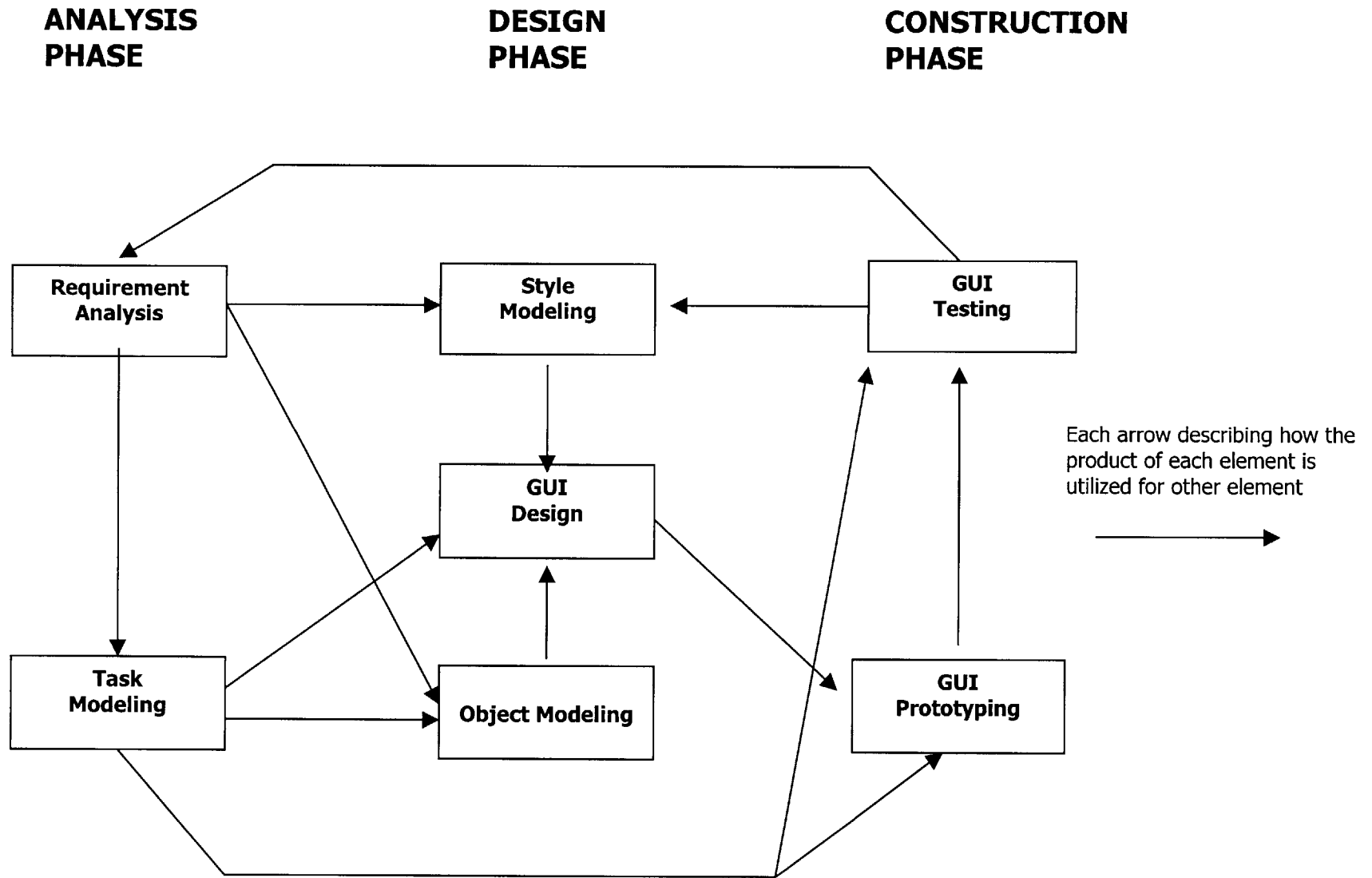


Figure 3-2 Collaborative Graphical User Interface

3.1.2 Collaboration Model

Collaboration is the process of shared creation: two or more individuals with complementary skills interacting to create a shared understanding that none had previously possessed or could have come to on their own [Schrage, 1995].

Collaboration includes all of the following elements: jointly developing and agreeing to a set of common goals and directions; sharing responsibility for obtaining those goals; and working together to achieve those goals, using the expertise of each collaborator.

The author describes the collaboration as the process of creating value by integrating all the knowledge and skills of collaborators into goals that would not be possible to achieve if the collaborator were acting alone.

Knowledge of team collaboration came from the author's first-hand experiences in the ieCollab project. In this project, the author had the experience to understand the idea of collaboration with its strengths and difficulties. The author tries to understand these strengths and difficulties, implementing the strengths and eliminating the difficulties for constructing CGUID.

In the CGUID, the author proposes collaboration of team member during each high-level design phase. Each team should address and develop the solution by communicating and filling the gaps from each team. In every phase, the team should work on each low-level design elements in creating desirable products that support the following low-level design element. The author predicts that by working together and by integrating knowledge between each collaborator, each of the problems with low-level design elements can be eliminated.

In Table 3-3, the author proposes the structure of the collaboration team according to phases.

Table 3-3 Collaborating Teams

Analysis Team	Design Team	Construction Team
Project Manager (Leader)	Interface Designer (Leader)	Developer (Leader)
Interface Designer	Graphics Designer	Interface Designer
Developer	Developer	Document Specialist
User	User	Tester
		User

Besides the software development process and team collaboration, the author uses other considerations for developing the idea of CGUID. The author uses incremental software development model and the user-centered design model will be discussed in the following sections.

3.2 Software Development Model

This thesis introduces two commonly used software development models: waterfall and incremental.

CGUID is included as the incremental development process instead of waterfall. This is due to the fact that the CGUID reiterates until the software meets the business and user's requirements. The following is an overview of both methods.

3.2.1 Waterfall Model

The waterfall model is the oldest model of all lifecycle models. Although it has many problems, it serves as the basis for other development models. This is due to the fact that it has the fundamental steps of software development: analysis, design, construction, and testing.

The waterfall model is the process of incrementing from smaller design phases into another smaller design phases, from the analysis phase until the installation phase.

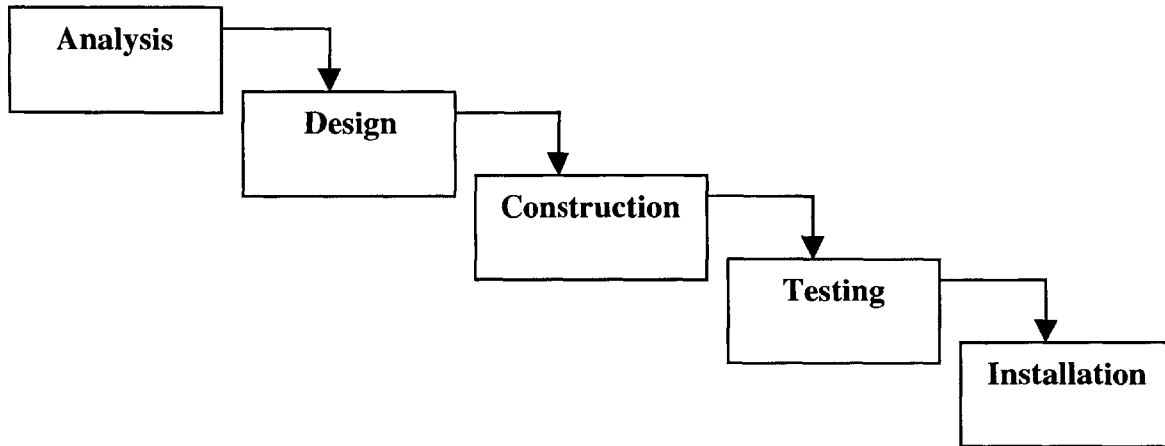


Figure 3-3 Waterfall Model

Source: Redmond-Pyle, *Graphical User Interface Design & Evaluation*, 1992

However this method has some weakness. They are:

- The user does not obtain anything until the software is ready to deliver. This method might result in the software being built but is not highly usable to its user due to lack of interaction between the user and the development team.
- There is little opportunity for problems encountered or experience gained during construction, testing or use to influence requirements or design. There is only one increment from the design phase to testing phase. Therefore, software developer might not be able to fix the problems because once they are realized, the software development is almost over.

Since it is done in one increment, the waterfall model works well with projects that are well understood but complex. The development team can benefit from tackling complexity in an orderly way. In addition, it works well when quality requirements dominate cost and schedule requirements. Elimination of midstream changes eliminates a huge and common source of potential errors [McCowell, 1996].

3.2.2 Incremental Model

The incremental model is an improved version of the waterfall model. The difference between the two models is that the incremental model supplies several increments for each stage in the waterfall model. Each method consists of a relatively self-contained unit of functionality which would provide the user with several benefits. The number of increments can be anywhere from three to fifty. The increments are prioritized, either: [Redmond-Pyle, 1995].

- To give the priority to the greatest business benefit and the least cost.
- To give the priority to the high technical risk on which other increments are dependent.

The analysis phase is performed across the scope of the whole system so there is an appropriate high-level direction and frameworks. The increments are then developed in priority order with each increment progressing through analysis, design, construction, testing and installation.

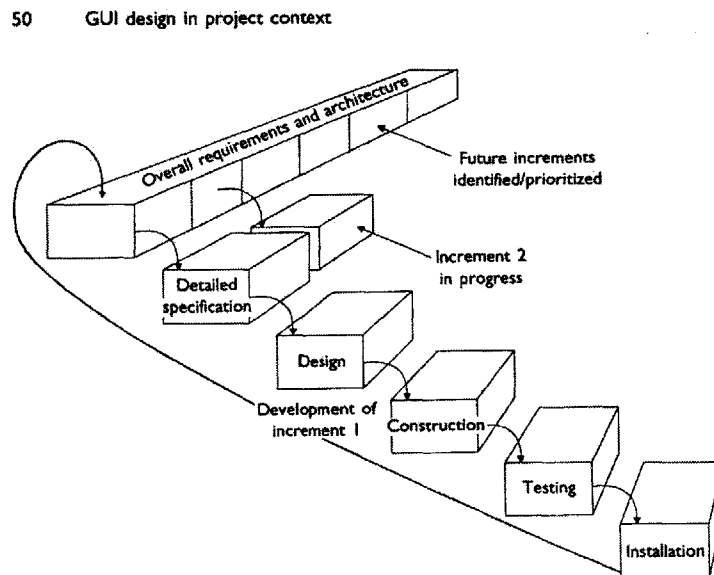


Fig. 4.3 Incremental development

Figure 3-4 Incremental Method

Source: Redmond-Pyle, *Graphical User Interface Design & Evaluation*, 1992

With the incremental model, the value added to the software development process includes the following:

- Because of the repetition, the elapse time between analysis and installation is shorter. Therefore the requirements specified in the analysis stage are less likely to change.
- There is feedback from the earlier incremental phases that can be used to improve the latter version.

CGUID uses waterfall model, with analysis, design, and construction phases. However, the three phases in the CGUID can be reiterated to achieve the desired output, satisfying the business and usability requirements. Therefore, beside being a waterfall model, the CGUID can be described as the incremental model.

3.3 User Centered Design

Pursuing usability means designing the system so that it is intuitive and convenient for the user to use, rather than developing what is convenient for the developer and the system, and then asking the user to adapt to the system.

Trying to achieve high usability for the end user has three important implications for the design process [Redmond- Pyle, 1992]:

- The designer needs to understand in detail who the end user will be, what task they will perform and what their specific usability requirements will be. Without this information the GUI design cannot be organized around the users.
- End-users play an active role in design team throughout the analysis and design process. It is not sufficient to interview some users at the beginning for their requirements and to ask them to review the design once its is complete.
- The designer and end users jointly evaluate the usability of a proposed design as early as possible and modify the design in the light of this feedback.

All of these, the emphasis on users' tasks, user participation and user evaluation are referred to as a user centered design.

3.3.1 Contrast with system centered design

System Centered design works out from the heart of the system, and come to the end user last. The typical design sequence is as follows [Redmond, Pyle, 1992]:

- Specify user requirements which are normally actually business requirements concerned with functionality, with little mention of human being who will be the end users.
- Specify the heart if the system.
- Specify system function.
- Design a user interface to put on top of the system functions, to allow the user to invoke them, and to provide the input the system needs and to format system output.

In CGUID, the author values the involvement of the user in every phase of the software development model. Therefore, the user involvement is included in each stage of development process and the graphical user interface will reflect the user's proficiency.

3.4 Summary

Collaboration Graphical User Interface Design (CGUID) is a method for utilizing different parts of software development phase, interacting, and reiterating them to create a product that satisfies the usability requirements.

CGUID is a team collaboration solution. Collaboration allows the team member to exchange information and to achieve output that is hard to achieve with the collaborators working individually.

CGUID is the combination of the software development phase in a collaboration model that allows team member to create an effective graphical user interface design.

Besides these two concepts, the author implements the increment model and user-centered design in CGUID. The increment model allows iteration until the graphical user interface has fulfilled the business requirement. The user-centered design, on the other hand, emphasizes user involvement in the GUI development process.

The following three chapters discuss the high-level design components - Analysis, Design, and Construction Phase. Each of these phases has team collaboration and software development processes.

Chapter 4

Analysis Phase

4.1 Introduction

The analysis phase has two low-level design elements: Requirement Analysis and Task Modeling. The Requirement Analysis process generates the user profile and usability requirement (Figure 4-1), while the Task Modeling process produces task scenarios and task models (Figure 4-2).

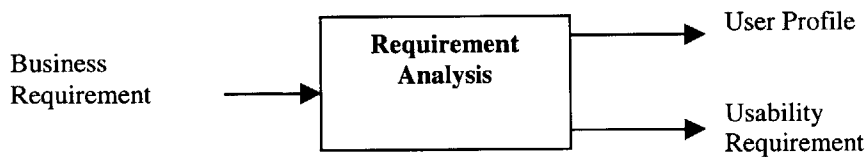


Figure 4-1 Requirement Analysis

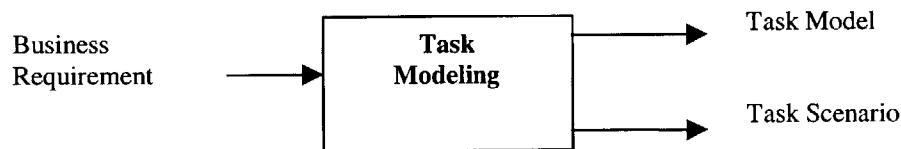


Figure 4-2 Task Modeling

The business requirements from the business idea specify the requirements for the GUI application developed. Examples of business application include a design of an automated system for company's inventory tracking or an application that handles hotel's room reservation. The user profile is a subset of the user population which share similarities in terms of their system usage and relevant personal characteristics. The usability requirement answers what the user expects from this new system. Should it be easy to use or easy to learn?

Besides the user profiles and the usability requirements of the Requirement Analysis, the analysis team should also identify the task scenario and develop task model. A task scenario is a task to be perform. A task model is an abstract structure showing many possible variants and subtasks. The difference between the two is that a task scenario is a single concrete instance of a task, with specific initial circumstances, input and subtask performed [Redmond-Pyle, 1995].

Section 4.2 describes collaboration occurs within the design team. It describes who the major stakeholders are and what their contributions in the collaboration process are. The following two sections (Section 4.3 and Section 4.4) describe how the process of developing the analysis phase products: user profiles, usability requirements, task models, and task scenario.

4.2 Collaboration in the Analysis Phase

The interface designer, developers, project/business managers, and potential users are the stakeholders in the analysis collaboration. The interface designer interacts with users to get information for the user profiles, usability requirements, and task models. Developers ensures to the usability engineer and project manager that building the software is possible. The project/business manager supervises the analysis meeting with users. This manager behaves as moderator if the users are not technically compatible with both interface designers and developers. The users are ultimately the sources of information for user profiles, usability requirements, task models, and task scenarios.

The collaborators of the Analysis Phase are described in Table 4-1:

Table 4-1 Analysis Team

Roles	Description of Roles
Interface Designer	Designs the graphical user interface using the analysis result. Gathers preliminary data about the user, task, and the work environment populations.
Developer	Writes code that provides the functionality specified in the GUI and system requirement.
Project/Business Manager	Supervises the meeting with the client and behaves as a moderator if the end user is not technically compatible with the usability engineer or developer.

The collaboration process can be done in a meeting room or in the virtual meeting room using meeting management tools. However at this thesis, the author describes the meeting at the client’s work place which is known as “field study.”

Field studies are powerful ways to get useful data quickly. Field studies also allow the analysis team to see the context of people’s work, for example, as in the physical environment, the stress level, number of people, computer equipment, and existing system.

There are always surprises when the analysis team goes out and watches users in the field, and therefore, it is almost always worth the extra time and expense that field studies might entail. Many interface designers say that they cannot design a usable system if they cannot conduct at least one field study [Weinschenk, Jaman, Yeo, 1997].

In the field study, the analysis team with the users should collaborate in a meeting medium. Usually this meeting medium is a table with the analysis team and users sitting around it. This medium allows the analysis team and users to collaborate in every activity. These activities will be described in Section 4.3 and 4.4.

First of all, the collaborators should understand the business idea behind the application development. It is fortunate if the analysis teams are associated in some areas that the system might be involved in. The GUI application can be developed for the manufacturing, distribution,

finance and banking, and public sector. If the analysis team is customized with the necessary knowledge, then it is faster and easier for the analysis team to absorb and understand what kind of application that the users need. If the analysis team does not become familiar with the users job or main activities, it is the users' responsibility to compensate for the analysis team's lack of knowledge in designing the task model.

These are the guidelines for the interview that the analysis team should consider while they are taking the information from the users.

- **Simplification.** The end user should not simplify the description of his task. This might be caused because the end users might think that the analysis team does not understand about the tasks that the end user does. The end user should mention all the detail tasks step by step. Having the user simplify their task might lead to important tasks not being captured in enough details, thus changing them in design phase will be more difficult and time consuming.
- **Not Just Watch.** Sometimes it is tempting to start designing the new interface while the analysis team is watching the user use the old system. However, this is not the purpose of the interview and field studies. The analysis team needs to observe and collect data, not analyze an design. Because design can be done later on, which is also covered at the later part of this chapter.
- **Make the User Comfortable.** People are often nervous about being watched and interviewed. The analysis team shall introduced the team and explain what the team doing and why.

The manager shall supervise the whole meeting process. The users might not have the technical knowledge like the developer and interface designer. On the other hand, the developer or interface designer assumes that the end users are like themselves. During the process, the business/project manager becomes a moderator preventing miscommunication between technical and non-technical collaborators.

Finally, the users shall revise and comment on the work performed by the analysis team. The analysis team shall conclude the analysis process after all the results are available and the users agree with the user profiles, user tasks, task scenario, and usability requirements.

4.3 Requirement Analysis

4.3.1 Identifying User Profile

Human psychological characteristics such as attitude can not be changed. They just come in a package and the analysis team should be able to recognize these. The designer team cannot control the attitudes or level of motivation. The designer only can minimize the negative emotions of fear, anxiety, threat, boredom, apathy and to maximize job satisfaction. The attitude types can be positive, neutral, or negative.

The motivation types are divided into discretionary and mandatory. A discretionary user has an option to use the system. Therefore they need to feel immediately that the system will not take too long to learn. A discretionary user usually has low motivation. A mandatory user must use a computer as part of their job. The system must provide benefits from the immediate experience using the computer. A mandatory user who is highly motivated out of fear (losing their job, doing homework) need the reassurance that the system is not overly complex and will not be difficult to learn.

The education level of the user has to be explored. Has the user completed high school? Does the user have a college degree? Does the user have advanced degree in specialized areas related to system use? Education level is generally correlated with reading level. If much of interface is verbal, then it is important that the vocabulary and grammatical structure be at the level that can be easily understood by users. Users' reading levels can be objectively measured, but if this is not possible, reading level can be inferred from educational level.

Computer literacy is the next level after educational level. The analysis team can measure user's ability relating to the existing computer experiences and skills in the hardware and software environments. How familiar the users use the hardware (keyboard, mouse)? What kind of software application they use such as (meeting management, financial, or accounting package software)? Have the user use GUI before and if they have used the GUI similar to the GUI that will be designed.

Analysis team should know whether the user has the basic task knowledge of the business that the GUI would serve. For example if a user is a person who work for airline reservation system, he or she should know which airport an airline would operate, what is the pricing policies, and the common connecting cities between departure and destination points. This task experience is called semantic knowledge [Shneiderman, 1992].

System experience refers to knowledge of the particular language or mode of interaction of a given system. User may have been performing their job for years effectively by hand, but will be unable to perform the job when it is automated until they have learned the idiosyncratic language of the new system. For the example of the airline reservation system, system experience pertains to knowledge of special function keys on the keyboard, codes for airlines and airport, the syntax for entering a reservation, etc. This system experience is called syntactic knowledge [Shneiderman, 1992].

How does often the user use the system? The user might use a system 8 hours a day, 5 days a week. On the other hand, they might use a system once in a lifetime. These represent the extremes on the range of frequency of use. User may fall anywhere between the extremes, with some interesting variation, such as everyday but only 5 minutes a day. Frequency of use has profound implications for interface design because it affects learning and memory. Frequency of use affects system design in two ways. First, user who spend a lot of time on the system are usually willing to invest more time in learning, and therefore efficiency of operation often takes precedence over ease of learning. Next, frequent users have less difficulty remembering system commands between uses and efficiency of information often takes precedence over memorability. On the other hand, low-frequency users will be less able to remember interactive techniques between uses.

Turnover rate is an important job factor. A system that requires a great deal of training and is difficult to learn would not be appropriate for a job with a high turnover rate. The expense of training may outweigh the advantages of the automated system. When the turnover is low, a greater learning burden can be justified and will be perceived as a reasonable investment.

The task importance automated by a new system will influence how much of an investment in learning users are willing to make. For example, an electronic calendar might be perceived as very important to a secretary, a decision support system might be very important to an executive, and electronic mail might be very important to a manager. On the contrary, a word processor might be relatively unimportant to a manager, and a graphic package would not be important for to an executive. In the earlier example the user is willing to invest a considerable amount of time in training and learning, whereas in the latter example they probably would not.

Task importance and frequency of use are not the same. The task may have a high performance but be executed with a low frequency. For example, a sales executive might implement the market modeling and forecast system as the basic for strategic planning. The system might be used only once or twice a year, but it is extremely important tool to this user. Then, in spite of low frequency of use, this user is willing to expend a considerably amount of time to learn to use the software.

4.3.2 Identifying Usability Requirement

There are three main categories that the author wants to emphasize for the usability requirement; ease of learning, ease of use, and rate or errors by user.

It is important to note that there are frequently trade offs between the usability requirements. For example, errors can be reduced by having low information density and using confirmation dialogs, however, this will reduce speed of performance. If longer learning is permitted, then the speed of task performance may be increased by asking users to learn special shortcuts or abbreviations. The relative priority of different usability requirements needs to be established early in the project

- **Ease of Learning.** If a system deals with a traveler to book a hotel or car rental at the airport, the system should offer ease of learning. The user can be in a variety of computer skill levels. Even, the user might be in pressure in time. Being easy to learn, the GUI shall behave as a tutor for the user instead of behaving as a tool.
- **Ease of Use/ Speed of Performance.** Contrary to the ease of learn, the GUI which server individuals with higher knowledge and computer skills should offer faster

performance. The purpose of these type of GUI is to maximize the amount of task done given the specified amount of time. Therefore the GUI should behave like a tool, in fact an intelligent tool that enable to increase the productivity of a user.

- **Rate or Errors by User.** Emotional arousal is the feeling the user have while he or she intreact with the system. Emotional arousal affects performance in a U-shaped function. Little arousal impedes performances, a moderate degree of arousal improves performances, and a high level of arousal also impedes performances. For example, the users who feels anxious about their ability to understand computers and fear that computers are going to change or eliminate their jobs or to whom the computer appear to be dominating and inconsistent will learn less quickly and perform with more errors [Mayhew, 1992].

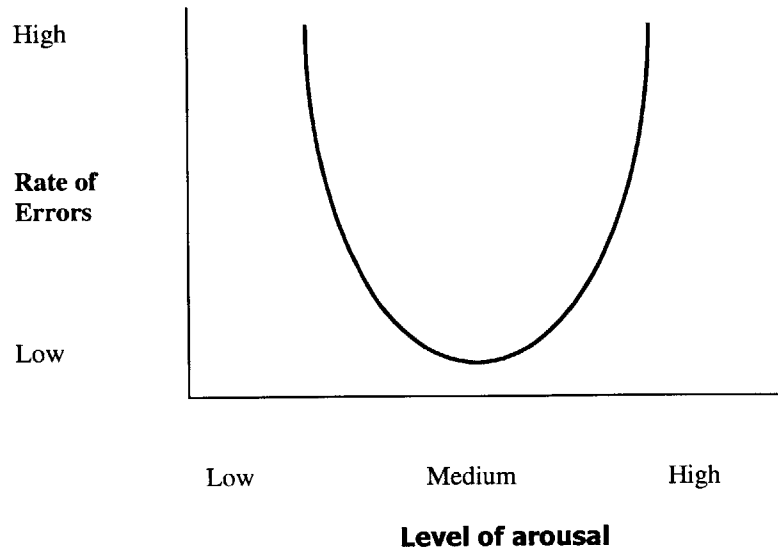


Figure 4-3 Arousal Performance Relationship

Source: Mayhew, D. *Principles and Guidelines in Software User Interface Design*, 1992

4.4 Task Modeling

Documenting the current task involves looking at how users do their work now in enough detail to provide the data the analysis team needs to design a new interface. Task analysis provides important clues to what the interface organization and conceptual design should be. Two important subjects in modeling task; user task and task scenario.

User task help the analyst to understand the world view of the user, in terms of identifying the objects in the system and the actions that people perform on these objects. Therefore, task analysis feeds the user object model, which in turn assists in the design phase.

Task analysis is the about learning what users do, not asking users to design interface. Task analysis only asks users for what they know best; their expertise on performing their work. The interface designer does not ask whether a user need windows. Through task analysis translates a requirement about a user who does not want to see three information at the same window into an interface design requirement for multi-windowed display.

4.4.1 Developing Task Model

Task analysis describes user task from the user's prospective, not the system functional or architecture prospective, and independently of any interface screen design concepts.

Task are shown as boxes, with lines showing how the task above is broken down to a set of subtasks. Thus, task X consists of subtask 1, 2, 3, and subtask 1 consists of two lower level subtask 1.1, 1.2 and 1.3. The diagram needs to be split into segments to make it easier to read. The diagram also needs to show sequence in which the subtasks are performed, or the logic of how the subtasks are combined to perform the task.

There is an issue about how far to continue subdividing the subtasks. Continuing to far greatly increase the size and complexity of the task hierarchy, and is not an effective use of time. Stopping too soon can mean that the important task requirements may not be identified [Redmond - Pyle, 1995].

The diagram shows the structure of task, but it is often not convenient for the diagram to contain all the detailed textual description of how the task is performed. Sometimes the diagram structure is all that is required, but where detailed text is appropriate this may be recorded separately from the diagram.

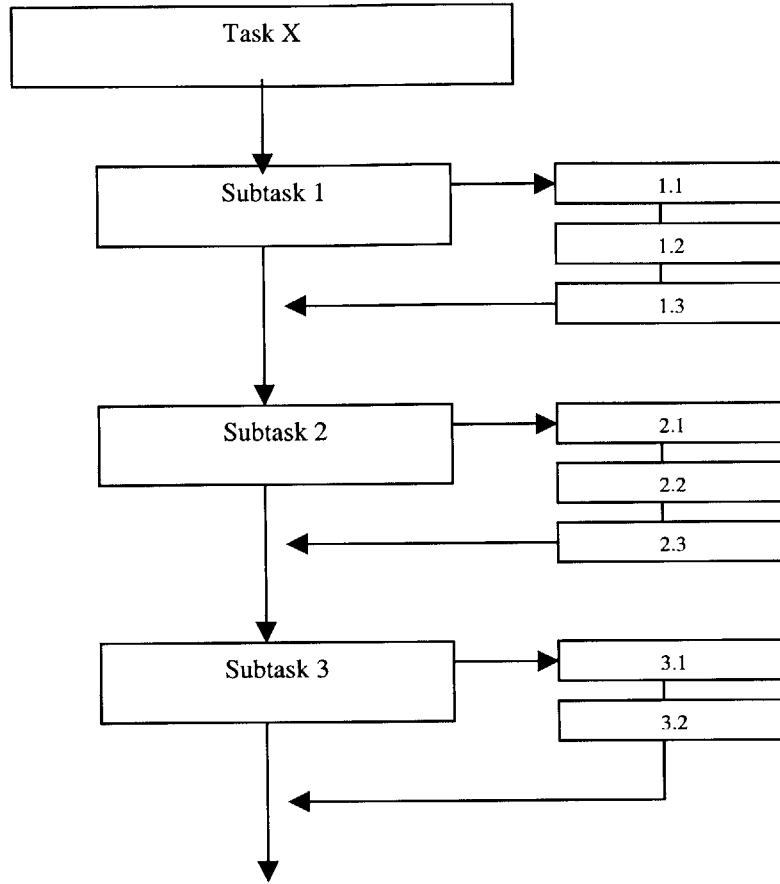


Figure 4-4 Task Model

Task table allows the analyst to capture and document all the data about each specific subtasks. The columns in the task detail table allow you to note how often user perform the subtask, what the user need to see to complete the subtask, what the user input, and a comments column where you can annotate problems with the current task flow an ideas for chages as the new interface is designed.

Task #	Task	Frequency	Display Requirement	Input Requirement	Comments
1.1	Locating Product	High	Warehouse ID	Product ID	Finding a product through warehouse in a region

Figure 4-5 Task Table

The analysis team also can enrich the task model by considering the contingencies which may arise while the user is performing task. What happens if some input information is missing or incorrect? What happens if the task ever interrupted to perform some other more urgent task? What happens if the user makes an error? For each contingency consider whether additional subtask

4.4.2 Developing and Modeling Task Scenario

Task scenario help to ground the development in reality. The scenario provides concrete examples of how the tasks will be performed, in a format that is easily usable by those involved in the GUI design and prototyping [Redmond-Pyle, 1995].

A task scenario is an outline of tasks and subtasks that describes how users will do their work. The purpose of the use case scenario is to aid in conceptual design. It must be describe how users will do their work with the new software so that the correct flow of screen can be developed.

A task scenario does not describe the user's task now, but the task they will perform when the new system is in place. It is easy when creating a scenario to fall into describing the current situation, but avoid doing that unless the new is exactly the same as the current.

The analysis team should build the task scenario from the user's point of view not the system's point of view. In order to match the way the users should be doing their work, the scenario must be a list of user tasks. If the analyst team want a system description in the task scenario, create a parallel scenario for the system.

The description of software actions is critical for software development, but it is a best a distraction for interface designer. This even can make the interface designer to digress from the real interface designer and move into designing the underlying software.

It is recommended that the interface designer and the software development group to work from the same task scenario. However both group should create two separate, but matching scenarios; one for the interface design and one for the software development design. This way,

the both interface designer and the developers are getting the same scenario, but each group can get the information it needs.

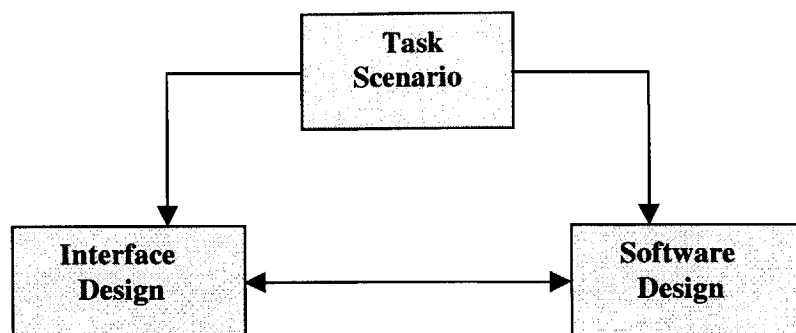


Figure 4-6 Task Scenario

Source: Weinschenk, Jamar, Yeo, GUI design Essential, 1997

4.5 Summary

Collaboration in the analysis phase employs the skills of analysis team (interface designer, developer, and project manager) and users to work together in creating the analysis products: user profile, usability requirements, task model, and task scenarios.

User profile is the list of characteristics of the individuals who will use the new application. Usability requirements are the requirements the user wants to see and feel while they are using the application. Task model and scenarios are the list of actions to complete a task.

The interface designer and users are the major stakeholders in the collaboration process. The interface designer and users collaborate in creating all the products of analysis phase. The user profile interview can easily be done by having the users filling in a profile form or by having them interviewed by the interface designer. The usability requirements and task analysis usually implicitly extracted from the result of interview.

Having the interview result ready, the developers ensure whether the result can be developed into an application. They also contribute some part of the user profile by examining the

computer system and type of application the users use in their workplace so that the interface designer might be able to grade how proficient the user with computer system.

The manager supervises the meeting between the analysis team and the users. He also becomes a moderator between the users and the analysis team if their technical backgrounds are not compatibly equal. The manager helps to explain the technical terms which are unfamiliar to the user. This will make the user feels that their involvement is valued.

After all products of analysis obtained, the software development process continues to the next phase, the Design Phase. The usability requirement is utilized in the Style Modeling. The task model and scenarios is utilized for Object Modeling, and the user profile is utilized for Object Modeling and Style Modeling.

Chapter 5

Design Phase

5.1 Introduction

Following the analysis phase is the design phase, which consists of Object Modeling, Style Model, and GUI Design. All these low-level design elements have the inputs from the analysis phase. The following are the low-level design element descriptions.

First is the Object Modeling. It receives usability requirements and user profiles from the Requirement Analysis. Object Modeling creates user object models which will be used for GUI design.

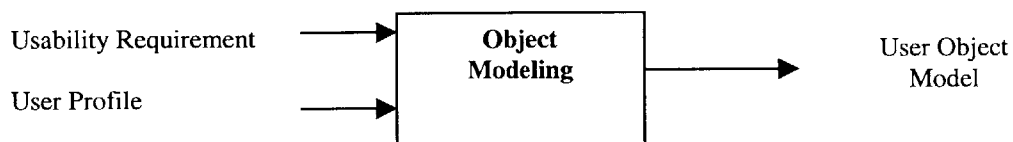


Figure 5-1 Object Modeling

The second low-level design element is Style Modeling. The objective of having Style Modeling is to create a consistent user interface style. Consistency style is supposed to increase the user productivity because a standardized style reduces what users need to learn and remember.

Consistency also reduces range of choices offered to interface designer in color, font size, window controls, etc. Finally, consistency increases the developer productivity because developers are applying and specializing standard formats rather than re-inventing new ones.

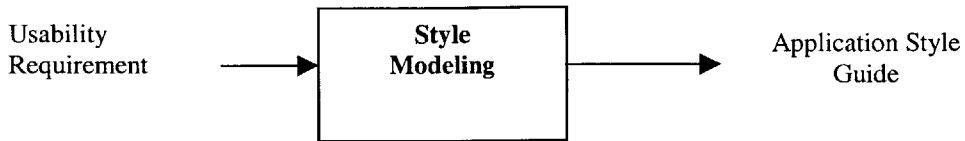


Figure 5-2 Style Modeling

Design of the GUI can be created from the style guide, task models, task scenarios, and user object models. The style guide defines the look and feel of GUI. It also defines the standard windows and sets of controls. The task model defines the scope and context of GUI design. The task scenarios are used when defining the detailed user interaction. Last, the user object model show what the user manipulates in order for achieving the task [Redmond-Pyle, 1992].

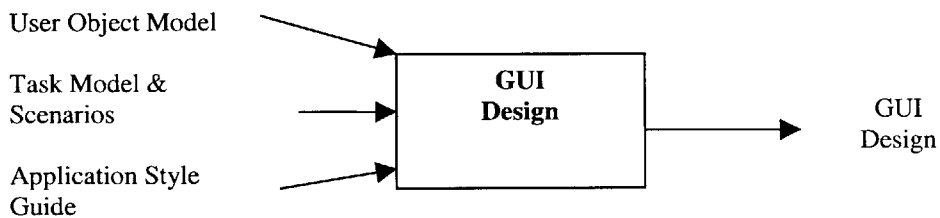


Figure 5-3 GUI Design

5.2 Collaboration in the Design Phase

The collaboration meeting can be held in a room equipped with a sketching tool. The sketching tools, either in the form of papers or whiteboard, are necessary apparatus for drawing the user objects and GUI design.

The collaborators consist of the role in Table 5-1.

Table 5-1 Design Team

Roles	Description of Roles
Interface Designer (Leader)	Translate the user’s task model and scenarios into major user object, metaphors and screen with control.
Developer	Evaluate GUI Design
Graphics Designer	Works with the interface designer in designing GUI

In Object Modeling, the interface designer works together with the developer to create the user objects from the task model and scenarios. The interface designer shall have all the task models and scenarios ready before the collaboration session with the developer. There are unsolved issues in the task model and task scenario, then the interface designer should clarify them with the users.

When developer collaborates with the interface designer to create the object modeling, the developer shall verify the object models so that they can be implemented for prototyping. Perhaps, the developer even can have larger contribution when designing the object model because he or she is the one who will make the prototype from this object model.

In Style Modeling, the users, an interface designer, and a graphics designer collaborate in defining style for the GUI. The styles usually are based with vendor style guide such as Microsoft’s Window, IBM’s Common User Access, or Macintosh style. The interface designer shall find out the styles that have already been used in most of users’ applications so that it is easier for the new application to interact with other users’ application.

Also, the interface designer also shall choose metaphors and verify whether the users are comfortable using the metaphors. Metaphors are the tools used by the interface designer to visualize the conceptual representation of user object and their associated actions. Desktop metaphors seen in the Apple Macintosh interface and the IBM Presentation Manager office are good examples: Common office objects (desk, file, folder, in box) are used in the interface

instead of their more unfamiliar system counterpart. The interface designer shall be able to get the metaphors that most likely favorable to the user.

In the GUI Design, the contribution of graphics designer is important because they can draw interfaces that might be interesting for the user. The graphic designer is valued for his or her ability to sketch the interfaces as the interface designer describes all the objects, attributes, and actions. The graphic designer can give suggestion to the interface designer in choosing the metaphors for the GUI design.

The interface designer and the graphic designer can use the sketching tools to draw the GUI design. They can use the paper media or whiteboard as the medium. The medium shall be easier to erase. There will be many updates during the task model and scenario reviews.

The developer then has to verify the design whether it can be implemented. The developer should look at the design before the users do. The user is the last person who decides whether the design is valid and ready to be prototyped. They can discuss with the interface designer if the design does not suit their intentions and needs. The iteration process in GUI design stops until the users agree to the design that users think have the usability requirement that they need.

In Section 5.3, the steps for creating the object model are introduced. The following section, Section 5.4, the steps for style model that match the users' need. Finally, Section 5.5 is the information about GUI design.

5.3 User Object Modeling

5.3.1 Identify the User Object

User object can possess attributes, relationships with other user object, and user object actions. The information gathered in the field studies has to be used to identify the user object. The first source is a task scenario. For example, do user talk about clients, project, and employee? Are

their tasks "create a new budget" or "update employee list" What the analysis team sees the users manipulate these objects while they are performing their tasks. These are the objects the interface needs to capture as the user object [Weinschenk, Jamar, Yeo, 1997]. To identify a user object effectively from the task scenario, these two question have to be asked; what is created or changed by this task and what objects are used or referred to in performing the task? [Redmond- Pyle, 1995].

Another source to identify the user object is from a data model. When examining a data model, the interface designer should identify the user object from the entities given. The entity that becomes a user object should have a simple primary key, which more called as fundamental entities. For example car is fundamental entities because it has primary key registration number.

However not all the entities become user object. The entities that do not become user objects are [Redmond-Pyle, 1995]:

- Entities which are used to describe some characteristic of a more substantial entity. In data modeling these are referred to as "characteristic entities", in that they are often created by normalization of repeating groups of characteristics.
- Entities which are use to express a many to many relationship between to more substantial entities. Such entities typically have a compound key formed form the keys of their master (in data modeling these are refereed to as associative entities as the model association between entities).

5.3.2 Identify Attribute and Action

Attribute is the collection properties that characterize the object. For example a person can have a name, an address, and a data of birth. The criteria to uses for determining whether a piece of information should be an attribute of a particular user object are whether it is useful to support a task, and whether it seems sensible to the user.

If the object create from the data model than the attributes usually include all the attributes in the data model. It is useful to relate the user object attributes precisely to this data model,

when the interface element will need to be connected to database data. For any object which has attributes from more than one entity, associate the user object with the relevant entities and relationships [Redmond-Pyle, 1995].

The user object action is task thought by the user as an action that can be performed in on a real wold object. For example delete, create, check validity, and etc. User object actions describe the behavior of objects in the system. They are the main means of specifying required system functionality [Redmond-Pyle, 1995].

Define each action in terms of the following; brief narrative description, any input, any output [Redmond-Pyle, 1995].

The example of the object, attributes and action is described in the figure 5-2

Object	Attributes	User Action Object
Checkbook	Bank name Checkbook name Account Number Start Date End Date	Start a new checkbook Name Delete Print

Figure 5-4 Example of object, attributes, and action

5.3.3 Develop User Object Model Diagram

User object model diagram is used to visualize the types of objects and the relationships between them. Relationship is a way of connecting both user objects. The notation is an extension of widely used entity – relationships modeling notation, and uses the symbols. The symbol and relationship notion are described in the figure 5-5.

Symbol
<ul style="list-style-type: none"> ▪ Zero to one ▪ Zero to Many ▪ One to Many ▪ Many to Many

Figure 5-5 Relationship Table

Use user object model diagram to show the user objects and relationships between them.

The object modeling most commonly used is dynamic modeling. Dynamic modeling is useful where there is significant state-dependent behavior, i.e. where the behavior of an object changes through time, depending on the state that the object is in [Redmond-Pyle, 1995].

Table 5-2 shows the type of dynamic modeling.

Table 5-2 Dynamic Modeling

Dynamics Modeling
Sequence Diagram
Collaboration Diagram
State Diagram
Activity Diagram
Implementation Diagram

5.4 Style Modeling

5.4.1 Design Principles

The eight golden rules recommended by Shneiderman (1992) are a good example of these fundamental user interface design principles.

Table 5-3 8 Golden Rules in Design Principles by Shneiderman

Design Principles	Meaning
Strive for consistency	Consistency sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens, and consistent commands should be employed throughout.
Enable frequent user to use shortcuts	Abbreviations, special keys, hidden commands, and macro facilities are appreciated by frequent knowledgeable user. Shorter response times and faster display rates are other attractions for frequent users.
Offer informative	For every operator action, there should be some system feedback.

feedback	For frequent and minor actions, the response can be modest, where as the infrequent and major actions, the response should be more substantial.
Design dialogs to yield closure	Sequences of actions should be organized into groups with beginning, middle, and end.
Offer simple error handling	As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should detect the error and offer simple, comprehensible mechanisms for handling the errors.
Permit easy reversal of actions	The system should allow the reversibility. This feature relieves anxiety, since the user knows that errors can be undone. It stimulates the user to explore the unfamiliar option.
Support internal locus of control	The experience operator strongly desire the sense that the are in charge of the system and that the system responds to their actions.
Reduce short term memory load	The limitation of human information processing in short-term memory requires that the display be kept simple, multiple page display be consolidated, and window-motion frequency reduced.

5.4.2 Identify principle for the style guide

If the organization has its corporation style then the interface designer and software developer should follow the guidelines provided for the consistency. If there is no standard on the style guide then use the three most frequent style guide: Microsoft Windows, Macintosh, and OSF/Motif styles.

Most environment style guide includes the following;

- Design principles (Microsoft windows, Macintosh, etc)
- Standard control available in the environment (drop-down list, scroll bar).
- Standard menu headings (File, Edit, View, Help).
- Standard windows actions (Copy, Save, Delete).
- Standard highlighting and selection rules.

What if the user’s company does not have standard? This is a good time for the designer team to create a new standard. However this might need an additional effort because the style has to be apply for several systems established in that company.

5.4.3 Define Window Hierarchy

The interface designer should define standard window layout for the system, including the style of controls to be used on each type of window (e.g. pull-down menu, push button, and mixture). Then the design team should organize this type of window to window hierarchy. For each window class define the standard behaviors which be available (e.g. Open, Save, Minimize, OK, Cancel, Help) [Redmond, Pyle, 1995].

The standard types of windows [Redmond-Pyle, 1995]:

- Basic window has a size, a title bar, and a system icon.
- Modeless window has a menu bar with drop-down menus for File, Edit, and Help;
- Primary object window is a modeless window used to display the main view of an object. It has a standard menu bar with File, Edit, View, and Help.
- Secondary window is used to display and edit more detailed information about an object, or a component object. It adds an Edit menu option. It is dependent on a primary window. If the primary window opens a different instance of its object, the content of the secondary window changes to display this instance.
- Container window is a window used where the viewed object is just a container. The uses has little interest in the container, but is interested in viewing and manipulating the contents, moving objects in and out of the container, etc.
- Modal Dialog is used to capture additional input to perform a window action (such as number of copies to be printed, or confirmation of a deletion).
- Main window has a login dialog and a set of icons, each of which represents a dependent primary object window.

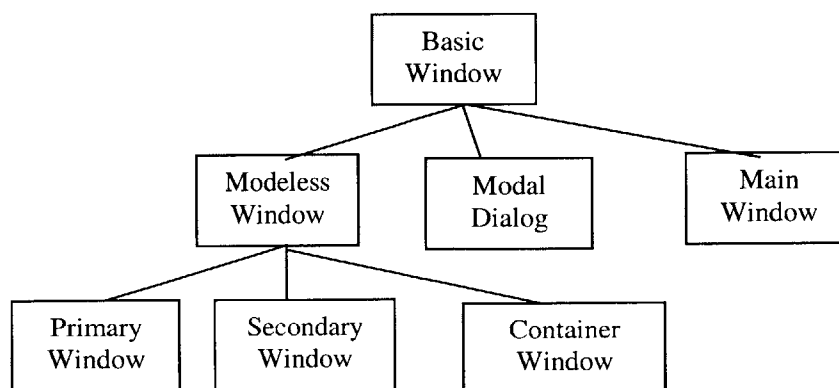


Figure 5-6 Window Hierarchy

5.4.4 Selecting Metaphors

Metaphors are the tools used to link highly technical, complex software with the user's everyday world. They are visual and conceptual representation of major user objects and their associated actions.

In choosing metaphors, it is recommended to choose the style that is familiar to the user itself. For example if the GUI application is the scheduling tools and have a list of name alphabetically then the GUI representation should metaphors as a roledex card. The roledex card is a familiar object to user thus having this presentation makes the user comfortable using this type of metaphors.

In addition, by having the metaphors such as roledex card, then the user will make assumptions about functionality and interaction. They assume that they can move alphabetically and be able to change or add information [Weinschenk, Jamar, Yeo, 1997].

Another part in choosing metaphors, the interface designer should take into consideration of actions taken upon the objects. For instances using a roledex as a metaphors implies the use of cards as an object, as well as actions such as moving forward and backward alphabetically.

5.4.5 Validate Style

Once the application style guide has been created by the design team, the developer and the user are necessary to review this style guide if it is applicable to them. If the one of these roles do not agree on the particular style then the designer team should evolves the design.

5.5 GUI Design

5.5.1 GUI design product

Window is the means for the user to interact with the objects and methods in the system. The window has a frame with its boundaries and one or more panes. The frame contains variety of

control and serve to execute functions. Each pane contains controls that enables user to interact with the object.

Control as part of the design product has three main functions.

- Allow the user to perform actions
- Allow the user to see the state of object in the system
- Allow the user to enter data

Control can have two or three functions. Each GUI environment has specific standard of control. However most of the standard control such as menu, label, push button, option button, and scrollable list are available in most of the GUI environment.

Window action is the window function occurs if the user performs an action on control. For example clicking a button, editing a text, or dragging an icon.

Window navigation is concerned with how the user opens new windows and transfers focus from one window to another. The designer team needs an overview of window navigation to plan how the user will be able to perform tasks requiring more than one window. A window navigation diagram shows a number of windows and the navigation paths between them. An example for window navigation is when the user prompts the system to open a file. Then the system would give a new dialog window asking for the user to perform a task on it. After the user has selected the file, then the view return to the window before [Redmond-Pyle, 1992].

5.5.2 Defining Windows

The design team should manage the view of the windows. The view of the window can be related to the user objects. Each window might consist of one or more views of objects. However, a window usually consists of one object except for multi-object task then each window might have more than one object.

The designer can select the control for each window. The control of a window is related to the attribute of the user object. The guidelines for the control in the windows to avoid errors [Redmond-Pyle, 1992]:

- For a discrete set of attributes, the control should present the valid values only.
- For the attributes that cannot be changed, the control should be not editable.
- For the attributes that are not valid, the control which represents it should be disabled.

To ensure the well-defined behavior, the design team should identify the windows dependencies. The guidelines for modeling the window dependencies [Redmond, Pyle, 1992],

- If a window is closed, is there any windows affected by this?
- What windows should be open before I can open this window?
- If I chose to select a different object instances in this window, are there any other windows affected?

5.5.3 Developing Task Scenario as Window Action

In case of the task scenario involves more than one object, the user has to do the task by moving from one object to another. If each object associated with a widows then in there will be a number of windows being opened and utilized even only for accomplishing a single task. Window navigation design is concerned with defining the path how moving from one window to other windows.

For each task scenario involving two or more objects, the design team shall review the relevant windows and navigation paths on the window navigation diagram. Using the diagram, the team can explore and evaluate alternative navigation paths with the user. If navigation is not appropriate for a high volume task, the team needs to redesign the windows. Allocating views of two or more objects to one window can often reduce navigation. A multi-object window designed for a specific task is often less suitable for use in other tasks than a single-object window.

Related objects for each user objects are shown in the user object model. For each related objects consider its usage in tasks to assess whether the user:

- needs to navigate form the related object to the current object.
- needs to navigate form the current object to the related object.
- needs to navigate explicitly in both directions.
- does not require direct navigation.

Then after this, the design team can define the window actions to perform all navigation, and show the planned navigation [Redmond-Pyle, 1995].

Once the design team has defined the object window, then the next step is to define the modal dialogs. When the current window is a modal dialog, no other windows in the application are accessible. When the modal dialog is closed, the user is returned to the window form which the modal dialog was invoked.

Some user object actions are best implemented as a window action sequence using a modal dialog. Three common situations are as follows:

- Additional information is required to perform the user object action (e.g. input of parameters, selection of a target object from a list).
- The user object action involves two or more steps, which must all be performed together before other work can be done.
- An explicit confirmation or authorization step is required for an action.

The following needs to be for each modal dialog [Redmond-Pyle, 1995];

- Identify a suitable window class from the window hierarchy to use as a basis for this one.
- Define the full information content and controls for editing or display.
- Define the window actions.
- Consider how the dialog fits into the window navigation diagram.

5.6 Summary

Design phase consists of Object Modeling, Style Modeling, and GUI Design. Each of the design steps receives input from the Requirement Analysis and Task Modeling. Object modeling takes the task model and user profile. Style Modeling takes the user profile and usability requirement. GUI Design also takes task model and waits for the output of Object Modeling.

The products of the design phase are object models, style guide, and GUI design. The final product of the Design phase is GUI design.

The design phase takes the contribution of an interface designer, a developer, a graphics designer, and users. The interface designer shall come with the user object, attributes, actions, and metaphors by reviewing the task models and scenarios. He or she collaborates with the developers for the object modeling and collaborates with graphical designer for the style guide.

In producing the GUI design, the interface designer collaborates with the users, developers, and the graphics designer. They shall decide how the window system would look like, how each window navigates to other windows, and the object on each window. The user shall be the last person who decides whether the design is acceptable to them.

The recommendation for the collaboration space is a meeting room with the sketch tools. The tools can be in the form of paper or white board that allows frequent updates and redraws.

The next phase is the construction phase. The GUI final design will be implemented in a prototype tool that will be a GUI prototype.

Chapter 6

Construction Phase

6.1 Introduction

This last phase of CGUID is the Construction phase which consists of GUI prototyping and GUI Testing.

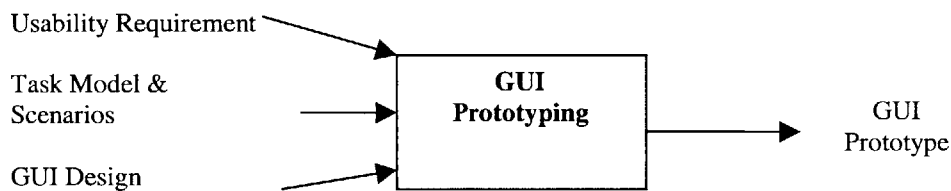


Figure 6-1 GUI Prototyping

GUI Prototyping takes the usability requirement, task scenario, and GUI design as input. The product of GUI Prototyping is GUI prototype.

The GUI Testing inputs GUI design, GUI prototype, and usability requirements. It produces evaluation report that summarizes how the design is evaluated. GUI testing is the method for finding problems as many as possible, that do not agree with the usability requirements specified in Requirement Analysis.

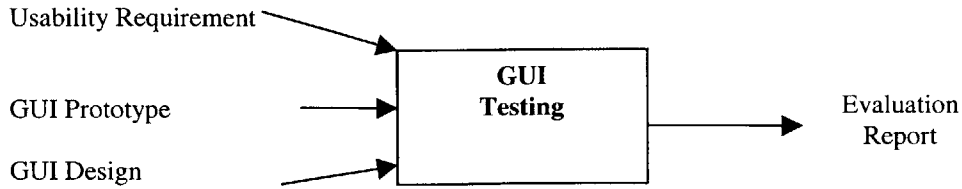


Figure 6-2 GUI Testing

6.2 Collaboration in the Construction Phase

The Table 6-1 shows the collaborators of construction GUI

Table 6-1 Construction Team

Roles	Description of Roles
Developer (leader)	Develops the prototype
Tester	Tests the prototype
Interface Designer	Ensures the usability requirements
Document Specialist	Ensures the standardization in prototyping

The collaboration shall occur in the meeting room with the prototype tools. Prototyping tools are used for creating the GUI prototypes. Usually, the prototyping tools come in the forms of computer application.

Teams in the construction phase focus on the development of GUI prototype. Before the collaboration starts, the developer shall verify that all the GUI designs are clearly understood. The developer shall also consult with the interface designer for the unclear GUI designs. Depending on the amount GUI prototype work, the numbers of developers can be increased up to point where the number of developer will not interfere the development of GUI. If the construction team happens to increase the number of developers, the developer who has been collaborating for all three phase shall be the team leader of the prototyping team. He or she has the experiences of the last two phases and the usability requirements specified by the user.

In GUI Prototyping, the lead developer interacts with the interface designer and the users to review the design aspect; style, GUI design and prototyping tools. Then, the lead developer collaborates with the other developers (junior developer) to start prototyping. The document specialist comes to play to push the standards in prototyping. The prototype should follow the standard rules. For example, the codes have to follow IEEE standards or what kind of documentation the developers should include inside their codes.

In GUI Testing, the developers, the users, and the tester shall discuss the prototype for evaluation. The process starts when the tester receives the GUI prototype from the developers.

The tester shall do the testing process first before handing it in to the users for testing. The tester shall create test case and record the test log during the testing process. The outcomes of the testing shall be reported to the developers and this outcome can go back to the involvement of interface designer if necessary. The iteration stops until the interface designer is agree to have the prototype being tested by the users.

6.3 GUI Prototyping

6.3.1 Define Prototype Objective

The following steps need to be done in the prototyping iteration [Redmond, Pyle, 1992]

- To validate the style guide.
- To reach agreement on the GUI supported for a particular task.
- To explore the appearance and behavior of a single complex window.
- To satisfy a specific usability requirement, such as making some part of the interface sufficiently simple that it requires no user training.

Assessing the acceptability of the style guide might be achieved by constructing a throwaway prototype to demonstrate the interface style of all the major windows in the windows hierarchy.

Providing a high usability level for a particular requirement may require a vertical prototype covering only a small part of the scope but implementing it fully [Redmond-Pyle, 1995].

6.3.2 Choose the Prototype Tools

In choosing the prototype tools, the designer team shall see if the interface can be built using the particular prototype tools. For a particular media such as voice and video, the prototyping tools has to be able have the necessary degree of integration.

It is expected for the GUI to run with other application running in the user workstation. Another aspect is platform. Some prototyping tools support multiple platforms.

The number of GUI tools in the market place is staggering, and not all them will survive. Check that the vendor is financially stable and provides good support for the product.

6.3.3 Build the Prototype

It is suggested that the team size for building a prototype be quite small, no more than five software developers in all. Having the large number tends to make the team fragmented and the communication become problems. It is important that at least one of the developers is skilled in the use of the chosen prototyping tools. Skilled developers can increase the pace of development both by their over the development efforts, and playing a technical authority roles [Redmond-Pyle, 1995].

6.4 GUI Testing

6.4.1 Identify the scope

The first step in testing is to plan the scope of the GUI testing. The components need to plan includes medium, interaction, and participants.

GUI testing include tests on GUI Design and GUI Prototyping. GUI design is usually done in paper while GUI prototype usually is done in computer media. Therefore the testers choose what medium that they wants to do; paper based during the design or computer prototype.

The user also can be included in the process of GUI testing. The tester team can have the user to interact with the GUI. Related to the interaction with GUI, the user's interaction can be divided into two; low interaction and high interaction. Low interaction is to have the user not interacting with the tester team during the process. Low interaction enable the tester team to see how the user interact without guidance, thus make the tester team see how the thinking of a user while he or she is interacting with the GUI. On the other hand, high interaction let the user has interaction to tester team. The benefit using this method is tester team can have understanding of the problems in GUI while the user is executing testing.

The users or participants can be chosen by evaluating their profiles in Requirement Analysis phase. In the user profile, the tester knows the capability of each user. By reviewing the demographics or profiles of the user, the test team can decide on the demographics of the particular test group.

6.4.2 Preparing the Test

After the medium, interaction, and participant have been decided, tester team should develop the test case scenario. A scenario describes the actual task that the participant will through during the test.

After preparing for test case scenario, the tester should run the pilot test. The pilot test is using prototype that will be used by the participant during the test. During the pilot test tester team will check out the equipment, the prototype, test scenario, and timing. If there are problem arise before the pilot test, maybe because of the prototype or the scenario, tester team can address these problem before handing the test material to the participants.

6.4.3 Develop the Test Result; Evaluation Report

In the evaluation report, tester team should not only be copies of testing logs. Tester team has to explain and prioritize the findings. Most of the time the test scenario are good way to organize the report. This is because tester team can discuss the problem related to usability requirement per task scenario. Each problem found in each task scenario, tester team should provide the recommendation to solve the problem so that design team can go back and fix the problem.

The goal of creating a final report are to summarize what happened and to provide advice and information that others can take action immediately [Weinschenk, Jamar, Yeo, 1997].

6.5 Summary

The construction phase consists of GUI prototyping and GUI Testing. GUI prototyping receives task model and GUI design, while GUI Testing receive usability requirement, task model, GUI design, and GUI prototype.

The collaboration starts with a process held in a meeting room equipped with prototyping tools. The prototyping tools are the apparatus for the making of GUI prototype.

The major stakeholder in the construction phase is the developer. The other collaborators are the users, the interface designer, the tester, and the document specialist. The number of developers can be increased according to the size of GUI prototype. However, the increased amount of developer shall not make the GUI prototyping process delayed or uncontrollable. It is likely that the more developers are involved in the programming, the more communication problem arises. Therefore, the lead developer shall forecast take the number of junior developers necessary for the project.

The process starts with collaboration of developers and document specialist in GUI prototyping. The developers develop the GUI prototype with the supervision of document specialist. The document specialist ensures that the developers shall follow a standard in prototyping GUI.

The next process is the GUI testing. In this process, the developer collaborates with the tester and the users. After the prototyping process has finished, the developers give the GUI prototype to the tester. The tester shall validate the codes from the developers. Tester is required to document what he wanted to test, when he performed the testing process, and how he did it. Finally, the tester asks the users to test the prototype. The user shall give the responses on the prototype and the feedback shall be evaluated by both the developers and interface designer. The iteration process on user testing stops after the user is satisfied with the GUI prototype.

The next chapter describes a development process of graphical user interface design for ieCollab project. ieCollab is a project at the Massachusetts Institute of Technology in developing a meeting management software. The author will describe the incremental development process of ieCollab's graphical user interface.

Chapter 7

Study Case: ieCollab

7.1 Introduction: Meeting Management Tool

A large development project usually includes more than one participating elements. To collaborate these elements, the project requires a tool for management so that these elements contribute the highest output in increasing the project value. In addition to the numbers, participating teams usually are geographically dispersed and culturally different, which may present additional difficulties for collaboration. Having these, is there a management tool capable to reduce and even eliminates the problem of collaboration?

A group of student from the Department of Civil & Environmental Engineering at MIT developed a tool using state-of-the-art, cutting edge technology that enables synchronous and asynchronous collaborations: ieCollab.

ieCollab will develop virtual team management and execution collaborative applications to be deployed via the Internet, to allow easy access from all collaborators. As long as all the computers are connected one person from any part of the world can communicate with someone on the other side of the world is just one “click” away.

The following are the three development phases for the first version of ieCollab; meeting management. The development process was started in the early January 2000 and finished by the first week of April 2000.

7.2 IeCollab Requirement Analysis – Study Case

ieCollab is an application for collaboration. The software enables distributed teams to collaborate for performing their work as well as addressing the issues that come along during the development of a project.

Field studies was not implemented in this software creation. The source of information for the requirement analysis came from the business manager proposal for MIT 50K competition. This is the type of future type of software mentioned in the section 4.2.1.

7.2.1 Identify the User Profile; Construction Team

ieCollab serves the teams for the necessity of meeting management. The software first targeted construction managers, structural designers, and architects for addressing the issues in the computer aided drawing design. However the usage of ieCollab can be expanded into meeting management tools for groups other than construction related users. For the simplicity in the further sections, construction manager, structural engineer, and architect are associated with construction team.

Construction team has the knowledge of using the computer software. The construction manager might have the experience using management tools, the design engineer might have a solution package for the calculating the structural design, and the architect might has use the drawing package for the blue print drawing.

If the construction team meet up together for the scheduled meeting, they do this because it is a must. Therefore this is the mandatory use.

The software is generally used for the meeting and addressed some issues that involves more than the two element inside construction team. Nowadays the use of fax machine and email are still frequent. Therefore the frequency of use this software is low.

Turnover rate would be considered lower because only certain people in the construction team use this software.

How important is the meeting management software for the construction team? At this stage the meeting management software create more value of shorter the meeting time and higher productivity. Therefore the task importance is high.

Table 7-1 ieCollab User Profile

Criteria	Result of Requirement Analysis
Mandatory/ Discretionary Use	Mandatory
Knowledge Level/ Computer Skill	High
Turnover rate	Low
Task Importance	High
Frequency of Use	Low

7.2.2 Usability Requirement

With the result shown in table 7-1, the analysis team shall be able to determine the usability requirement for that particular the user profile.

A mandatory user must use the software in order to communicate with each other. Therefore each element in the construction team would have to spend time to understand to use the program. However, the turnover factor is not high. Therefore, these individuals are the ones who often use the software and they have to invest some time to learn about the software. In addition, task importance is high means that this software have to be use efficiently to solve problems.

Therefore it is concluded that the software should behave as a tool to solve the problem. It tends to have higher speed in performance rather than guide the user step by step to solve the problem.

In conclusion, the analysis team should pass the information to the designer team that they should create an application that is easy to use, not necessary ease to learn, and have a high performance in speed.

If the ieCollab meeting management has achieved certain market share portion where most of the user has familiarity with ieCollab, then the usability requirement can be change into easy to learn instead of being easy to use. At this point, the user feels that he/she is necessary to use the product and want to learn how to use it.

7.2.3 Task Model and Scenario

In meeting management, the users are divided into two roles; the leader and normal user. The leader can be a meeting leader or workgroup leader. The leader has some abilities that a normal user cannot do, which will be explained in the table 7-3

There are two parts that play important roles in the meeting management; workgroup and management.

For the normal user, these are the tasks which are related to workgroup management are:

- List Accessible Workgroups.
- Search for Workgroups by name and by description.
- Request Workgroup membership.
- Accept or Decline a request to join a Workgroup.
- Create a new Workgroup.

After the normal user becomes a workgroup member, then he/she will have the capability of executing these following tasks.

- Relinquish Workgroup membership.

- List members of Workgroup.
- List scheduled Meetings for Workgroup.
- List and access previous Meeting logs for Workgroup.

If the user who has not become a workgroup, but he/she create a new workgroup, then the normal user become a workgroup leader. The workgroup leader has the capability to do the following task:

- (The leader can invoke all the use cases pertaining to Normal Workgroup Member except “relinquish membership”)
- Add new member to Workgroup.
- Accede to or Deny request for membership.
- Invite a user to join a Workgroup.
- Revoke Workgroup Membership.
- List / Remove old Meeting logs.
- Change Workgroup leader. (Only then can the leader relinquish his membership).
- Remove Workgroup.

Another part beside the workgroup is meeting. A meeting can be executed by a user and another user, a user and a workgroup, or two workgroups. A normal user can execute the following tasks:

- Schedule a new meeting.
- Set meeting agenda.
- Set meeting roles.
- Select a meeting template.
- Accept or decline invitation to join a meeting.
- Start a scheduled meeting.
- Request to join a meeting.

Once the normal user schedules a new meeting then he becomes the meeting leader. A meeting leader has these following capabilities.

- Cancel a scheduled meeting. (A meeting cancellation message is sent to the user. This will appear in the Invitation List and can be clicked away. Also the graphical representation (color or icon associated with that specific meeting changes)
- Change the agenda, participants and scheduled time for a Meeting. (Edit meeting information)
- Option to save meeting logs.
- In addition the leader has all the use cases associated with a Normal Meeting Member. (start a meeting is an example)

Also any user invited to a meeting can

- Start a meeting. A meeting can be started within a small interval of time before it is scheduled to begin (say 10 minutes)

The task scenarios / use case scenarios are outlines for the tasks and subtasks that describe the users will do the work. The purpose of the task scenario is to aid in GUI design. It must describes what the user will do so the correct screen can be developed [Weinschenk, Jamar, Yeo, 1997].

Each task described in the Table 7-2 and Table 7-3 can produce one use case scenario. Two examples of use case scenarios are "create meetings" and "edit meetings". As mentioned in the chapter 4 in Figure 4-4, the use case scenario that will be discussed is not related to the system scenario.

Another more complex scenario that involves other scenarios is edit meeting task. A meeting leader is the only one can execute the edit meeting task. These use cases are the following,

"Schedule Meeting" and "Edit Meeting" use case only part of all the use cases available for ieCollab. The use cases are needed to create the window action which is more explained in the next section.

Table 7-2 User Case for Schedule Meeting

Task Scenario "Schedule Meeting"	System Scenario "Schedule Meeting"
<ol style="list-style-type: none"> 1. The user clicks the "create meeting" button. 2. The user defines the name of the meeting. 3. The user fill the work group description. 4. The user click the OK button. 	<ol style="list-style-type: none"> 1. The system records the user ID and information about the user who click the "create meeting" button. 2. The system communicates with database and check whether the name of the meeting has been created. Return an error message if the name has been used. 3. The system waiting. 4. The system records the name and description of the meeting.

Table 7-3 Use Case for Edit Meeting

Task Scenario "Edit Meeting"	System Scenario "Edit Meeting"
<ol style="list-style-type: none"> 1. The leader clicks the "edit meeting" button. 2. The leader specifies the name of the meeting. 3. The leader sees the primary page of edit meeting. 4. The leader sees the users that are requesting to be included in the meeting. 5. The leader sees the workgroups that are requesting to be included in the meeting. 6. The leader sees the users that are online. 7. The leader sees the member of the meeting users and workgroups. 8. The leader sees all the workgroups available. 9. The leader sees all the users available. 	<ol style="list-style-type: none"> 1. The system identifies the user ID and find the meeting created by user. 2. The system lists all the meeting that has been created by the user and find the information of the meeting that has been selected. 3. The system returns the main page with the name of the meeting leader, name of the meeting, and the description of the meeting. 4. The system gets the information from database. 5. The system gets the information from database. 6. The system gets the information from database. 7. The system gets the information from database. 8. The system gets the information from database. 9. The system gets the information from database.

7.3 IeCollab GUI design

As stated in the usability requirement, ieCollab is presented as an easy to use software. The user tends to be discretionary users that requires guidance in accomplishing the tasks with ieCollab. Therefore the GUI should behave as a tutor that guides the user step by step.

At this GUI development, we skipped the procedure to create the data model. The GUI design basically done by task analysis. The GUI designers come up with several windows with their composition as well as their actions.

The properties and the layout of each windows came from the task model, while their actions come from the task scenario.

7.3.1 Main window

In the main window, the user expects to see all the activities easily. Therefore all the required object has to be shown in the main page. These required objects are on invitations, meetings, workgroups, and meeting logs. Also, the main page is the page that be able to navigate the user to several windows related to the user profile, workgroup, meeting, search, and help.

In Figure 7-1, the ieCollab required analysis team proposed the design of the main window. However there are some limitations in this design that induce the GUI designer to create a new design for the ieCollab main window. The previous design unables to GUI designer to place several numbers of buttons for the main windows. The new window has eight buttons. The second is the design limitation on

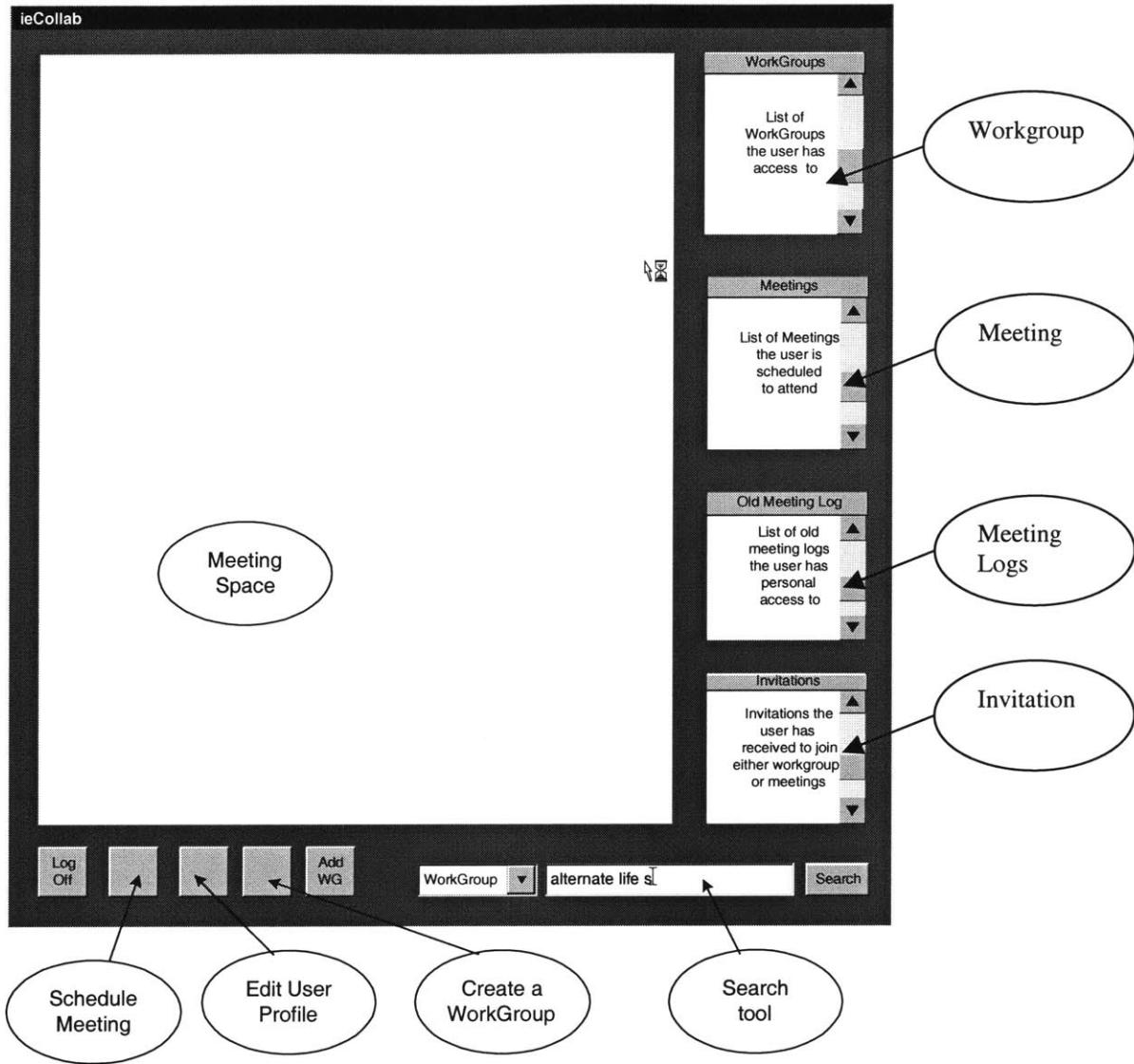


Figure 7-1 Requirement Analysis Recommendation for the Main Window

Source: ieCollab Requirement Analysis, Meeting Management Final Version, 2000

the layout in Java. At these stage the interface designer only use the Grid Layout and Box Layout provided in Java 1.2.2 JDK package. Therefore the designer team came up with the new design which is described in the figure 7-2.

Each button at the main window represents a task and use case scenario. The buttons on the main window are:

1. Exit
2. Help

3. Edit Workgroup
4. Create Workgroup
5. Edit User Profile
6. Edit Meeting
7. Search
8. Schedule Meeting

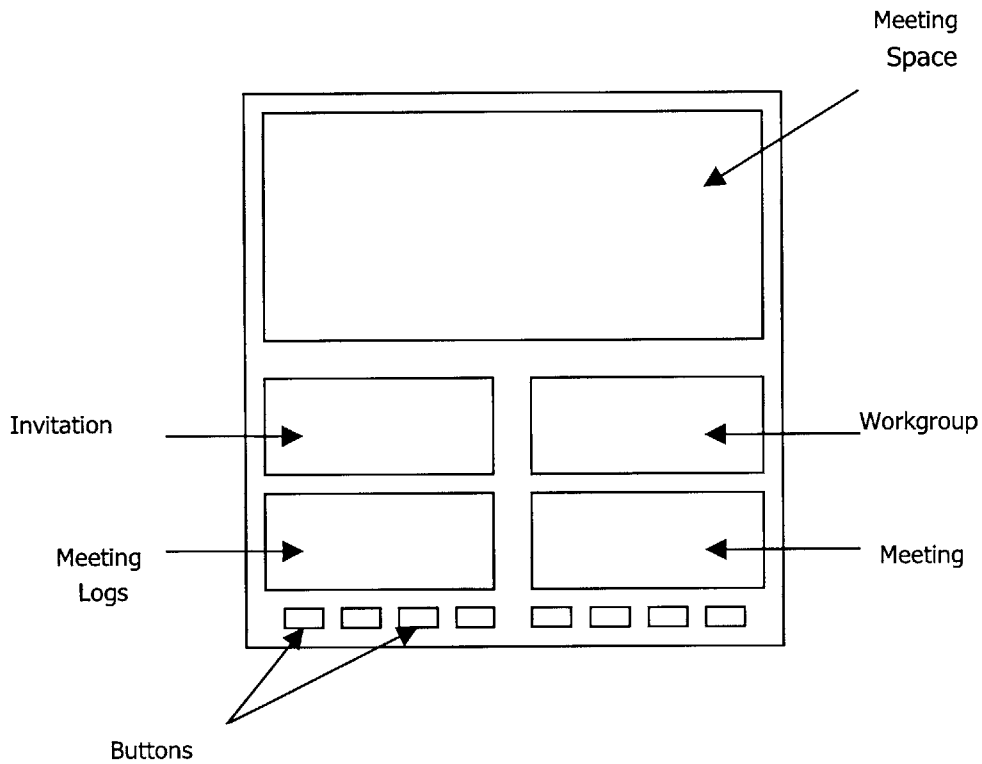


Figure 7-2 New Designed Main Window

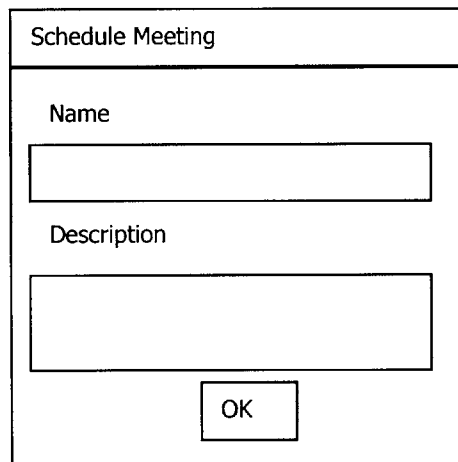
The main window is always the primary object window. It consists of several secondary object windows. The eight buttons will invoke the secondary windows. The error messages will be represented in the modal dialogs. In ieCollab development, the GUI designer only use three kinds of windows the primary object window, secondary object windows, and modal dialogs.

All the window actions should come back to main window as the primary object window. Another example for the secondary object windows are the edit meeting and schedule meeting.

7.3.2 Schedule Meeting

“Schedule Meeting” and “Edit Meeting” task scenario are described above at Section 7.2.3.

Schedule Meeting creates a meeting at the first time. It takes two inputs; name and description. Anyone can invoke this method.



The diagram shows a window titled "Schedule Meeting". Inside the window, there are two input fields. The first is labeled "Name" and the second is labeled "Description". Below these fields is an "OK" button.

Figure 7-3 Schedule Meeting Window Design

The schedule meeting window is prompt from the main window by clicking the “Schedule Meeting” button. The schedule meeting window ends if the “OK” button is clicked and the main window becomes active again. Unless the name of the meeting has been used before and the meeting with that name is still active, then the system will invoke a dialog window to prompt the user to enter another name.

7.3.3 Edit Meeting

“Edit Meeting” button on the main window prompts the edit meeting window. The user has to have the experienced of creating a meeting before. If he/she has not created a meeting then this user is not a meeting leader. The system automatically invokes a dialog window saying that the user did not create the meeting. If the user has created before then he is a meeting leader. If the leader has created more than one meeting and he /she click the “Edit Meeting” button,

then the secondary window pops off and show all the list of meeting that the user has been created. Once the meeting has selected then the edit meeting window will be invoked. Figure 7-4 will describe the design for the edit meeting window.

The window action for the edit meeting is the same as the schedule meeting window. The edit meeting window is invoked from the main window and back to the main window if the "Closed" button is clicked.

The inside this window, the list workgroups and users who requested to be included in the meeting. The leader has the ability to grant, reject, and review the profile of these workgroups and users.

The leader also can see the user who are online and who have been meeting members. If the leader decides to invite the users or workgroups then he/she can invoke these actions by clicking "Invite User" button or "Invite Workgroups" button.

Edit Meeting	
Meeting Leader	<input type="text"/>
Meeting Name	<input type="text"/>
Description	<input type="text"/>
User Requested for Meeting	<input type="text"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Workgroups Requested for Meeting	<input type="text"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
List of Online User	List of Invited User
<input type="text"/>	<input type="text"/>
List of All User	List of All Workgroup
<input type="text"/>	<input type="text"/>
<input type="button" value="Invited Users"/>	<input type="button" value="Invited Workgroups"/>
<input type="button" value="Update"/>	<input type="button" value="Close"/>

Figure 7-4 Edit Meeting Window Design

7.4 ieCollab GUI Development

7.4.1 Selecting the Prototyping Tools; Java Development Kit 1.2.2

In GUI development, the designer used Java version 2. The reason behind this because Java language is applicable for the internet and it can be used as Application Service Provider (ASP) Model. On the other hand, Java language can run in most of the available platform, thus enable to extend the ieCollab usage.

The Java Database Connectivity (JDBC) is another point added for considering the use of Java language. JDBC enables Java program to execute Standard Query Language (SQL) statement. With JDBC, Java program can communicate to with the SQL compliance database. Currently, most of the databases are SQL compliant.

Writing ieCollab code in Java language enables the ieCollab to run in most of the platform and communicate to large number of database management system.

7.4.2 ieCollab GUI

ieCollab GUI window properties is described in the following table.

Table 7-4 ieCollab Window Components

Name	Type of Window	Properties
Main Window	Primary Object Window	<u>Window</u> Invitation Window Workgroup Window Meeting Window Meeting Log Window <u>Button</u> Logout Button Edit Meeting Button Schedule Meeting Button

Name	Type of Window	Properties
Edit Meeting Button	Secondary Meeting Window	<p>Edit Workgroup Button Create Workgroup Button Edit User Profile Button Help Button Search Button</p> <p><u>Window</u> Workgroup Requested Invitation Window User Requested Invitation Window Online User Window Invited User Window Workgroup Window User Window</p> <p><u>Button</u> Invite Workgroup Button Invite User Button Update Button Close Button</p>
ScheduleMeeting Window	Secondary Object Window	<p><u>Text Field</u> Group Name Text Field Description</p> <p><u>Button</u> Submit Button</p>
Help Window	Secondary Object Window	<p><u>Text Field</u> Help Information</p> <p><u>Button</u> Submit Button</p>
Edit Workgroup	Secondary Object Window	<p><u>Window</u> User Requested for Membership Window Invited User Window Member Window Meeting Window</p> <p><u>Button</u> Update Button Close Button</p>
Create Workgroup	Secondary Object	<p><u>Text Field</u> Name of Workgroup</p>

Name	Type of Window	Properties
Search	Window	Description <u>Button</u> Submit Button
	Secondary Object	<u>List</u> Meeting, Workgroup, Invitation, etc <u>Button</u> Submit Button
		<u>Window</u> Result of the Search
Edit User Profile	Secondary Object Window	<u>Text Field</u> First Name Last Name Middle Name Street 1 Street 2 City State Phone Email
Login Window	Main Object Window	<u>Text Field</u> Login Name Password <u>Button</u> Submit Button Register Button
Register Window	Secondary Object Window	<u>Text Field</u> Login Name Password First Name Last Name Middle Name Street 1 Street 2 City State

7.4.3 GUI for Main Window, Create Meeting, Edit Meeting

The followings are the final prototypes for ieCollab. Figure 7-4 describes the main window, figure 7-5 describes the schedule meeting window, and figure 7-6 describes the edit meeting window. More of the graphical user interface presentation can be found in ieCollab's user manual.

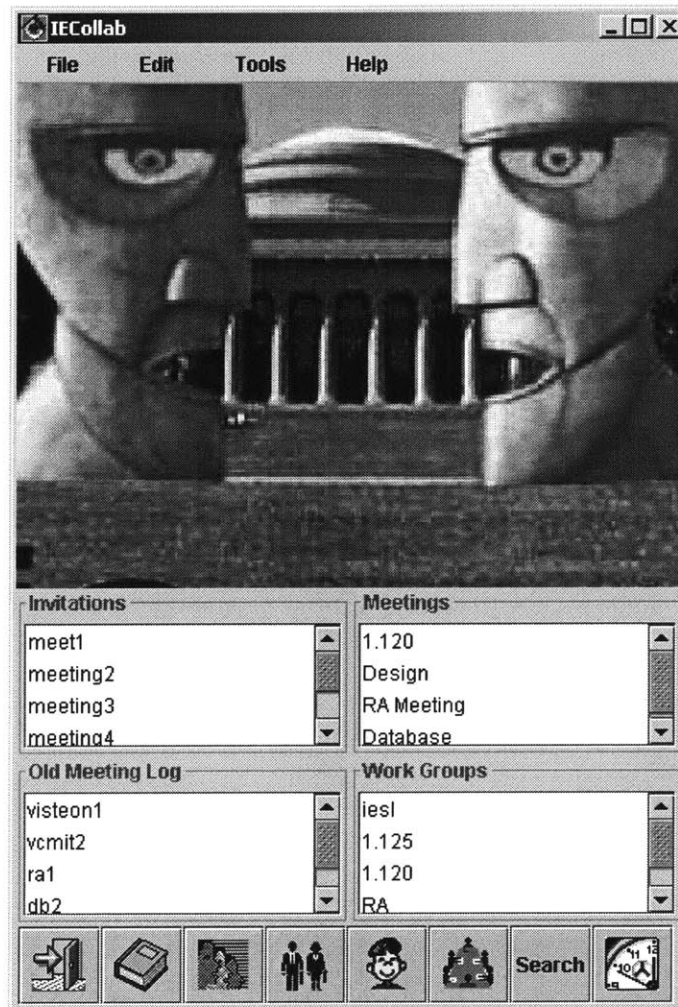


Figure 7-4 ieCollab Main Window



Figure 7-5 ieCollab Create Meeting Window

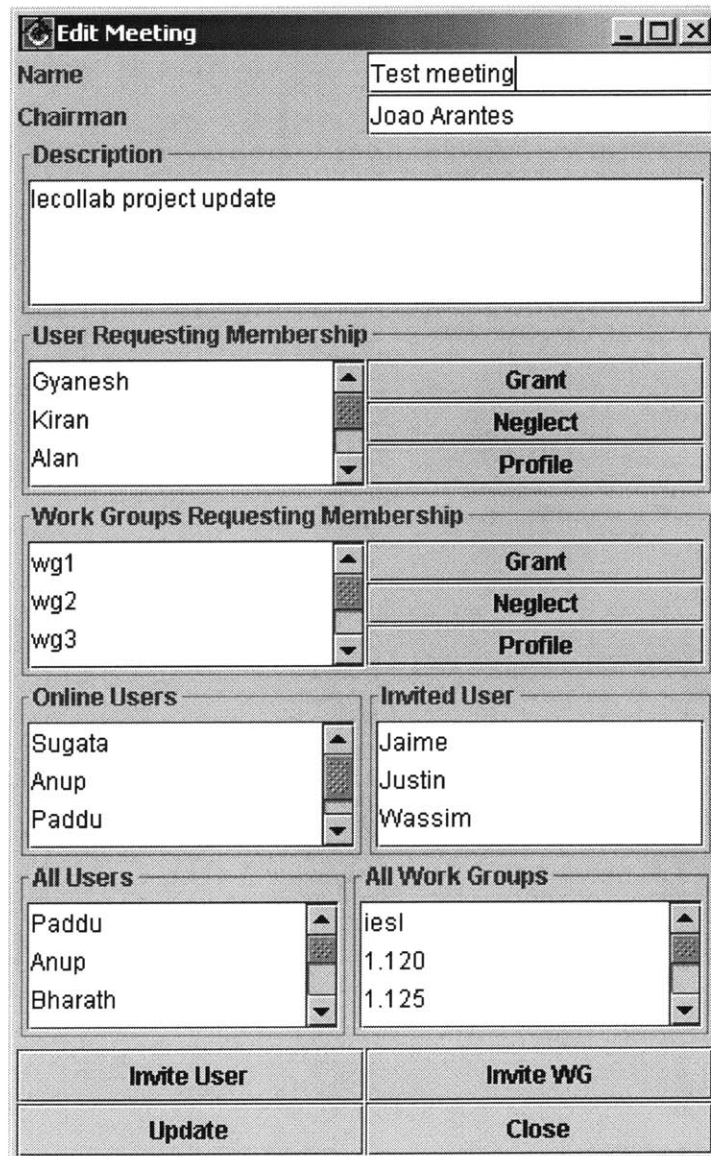


Figure 7-6 ieCollab Edit Meeting Window

7.5 Summary

As a member of the testing team the author tries to address the issue that come during the testing phase.

The first problem is the late design for the designer team thus the process of designing a test case being hold by this. The second problem is the collaboration with CICESE participant who has lost contact during the testing process. The last part is the lack of knowledge about testing while creating the test plan and test cases.

The solutions, proposed by the testing team, are in the followings. To address the late design, the analysis team should collaborate with the design team to build a consistent analysis and design. Thus if there is a problem, this problem can be addressed earlier so that it will not delay the design process. To address the collaboration with CICESE, the author believes that the number of participant in MIT should balance with the numbers of participants in CICESE. Since the numbers of team are different, the contribution of CICESE seems small compared to work done at MIT. Thus the MIT team value the team in CICESE differently. Due to different in valuing to each collaborator, then the collaboration does not work. To address the lack of information while doing the testing plan and case, the team concluded that the information session shall be done earlier in the Independent Academic Period. Equipping the team with the knowledge of testing earlier could make the test case process faster and accurate.

The other lesson was learned when the author executed his role as a programmer. The problem has to be solved are in the following. The first problem is none of the student wants to do programming. The second problem is that the difficult design to be implemented. To solve the first problem the student has be given an authority to assign the task to a particular student. At that point, the author did not see any other possibilities to increase the motivation of other student to do programming. To solve the second problem, the designer and the programmer should collaborate to design instead of just the design team. By doing this, there are possibilities that the problems can be found during the design stage and both designer and programmer can solve the issues. Another reason is to ensure that the design can be implemented from the programming point of view.

Chapter 8

Future GUI: Collaborative GUI

8.1 Introduction: Collaborative GUI

Collaboration is the idea of having more than one element meet at a medium to create value. The value can be in the form of creating new products or services. In ieCollab, teams collaborate to create meeting management software.

The idea of collaboration can be extended by changing different types of collaborators. Usually the collaborator are teams or humans. However, collaborative GUI introduces the notion of information as the collaborator, not humans.

The collaborative GUI is aggregation of several sources to establish a single GUI. The GUI can be in the form of applications or Web pages, while the sources of information obtained from the Web sites and databases.

8.2 Problems Data Collaboration

In this new era of Internet, more and more data are presented in the form of Web site. The publicly disclosed information can be retrieved as long as the user knows where to find them on

the Web. It is necessary to have data come not only from the database only but also the Web sites.

However, there are three problems concerning extracting data obtain from the Web. The first problem is structure of the Web pages, the second is searching for data from the Web, and the third is ensuring the data served the purpose of the user.

8.2.1 Context Issue

The Web pages are not structured exactly like database because they are not in the form of schemas and fields as the database system. The Web site is only text with the systematic design because there is no guidance for the author to create his or her Web sites.

Most of the documents on the Web are stored and transmitted in the form of Hypertext Markup Language (HTML). HTML is a simple language well suited for hypertext, multimedia, and the display of small and reasonably simple documents. HTML is based on Standard Generalized Markup Language (SGML), a standard system for defining and using document format [Bosak, 1997].

However HTML does not support the following criteria to make the Web site into database because of the following reasons [Bosak, 1997].

- **Extensibility.** HTML does not allow user to specify their own tags or attributes in order to parameterize or otherwise semantically qualify their data.
- **Structure.** HTML does not support the specification of deep structures needed to represent database schemas or object oriented hierarchies.
- **Validation.** HTML does not support the kind of language specification that allows consuming applications to check data for structural validity on important.

These are limitations for constructing the Web sites with HTML. Thus it is quite difficult for the Web pages to be used as a database.

8.2.2 Data Mining

If a user wants to see the last price of one particular stock, he/she might go to one of the financial information such as www.yahoo.com or www.quicken.com to find the following data. This procedure has to be done manually.

The information in the Web site is not explicitly presented and is put together among other information that is not necessary to see by the user. Thus this method is time consuming.

If one can imagine that he can have tools to find the query he needs in the Internet. This would reduce unnecessary tasks and increase productivity.

8.2.3 Data Content and Accuracy

Finding a data from the Internet is not a difficult task. However, to get accurate data that serves the purpose of the user is a challenge.

The information obtained from the web can have several meanings. Earning per share (EPS) can be yearly or quarterly. Therefore all the information from the Web is not completely has reporting the value that the user wants. This is not because of the data is not correct, but more to how the correct data given to the correct inquirer.

Another problem is translating the data. Companies that undergo merger process have problems in combining their database management system because their schemas are different. In addition, a project that consists of different elements that combine their works also need good connectivity for each of their work to collaborate effectively.

Nasa's Mars Climate Orbiter was lost because engineers did not make a simple conversion from English unit to metric, an embarrassing lapse that sent the \$125 million craft off course...

...The navigators [JPL] assumed metric units of force per second, or newtons. In fact, the numbers were in pounds of force per second as supplied by Lockheed Martin [the contractor].

Source: Kathy Sawyer, Boston Globe, October 1999, page 1

To find data with the right meaning is the purpose for the content mediation engine which will be discussed in the next section.

8.3 Technology for Collaborative GUI

These are technologies that are part of the solution to the three problems described previously. The first problem for achieving the uniformity of unstructured web is solved by implementing XML. The second problem in getting the data from the web is resolved by using web wrapper technology. The last problem is finding the correct data.

8.3.1 XML

XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some form of structure [Bosak, 1997].

A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents.

The HTML different from XML in the three major aspects [Bosak, 1992]:

- Information providers can define new tag and attribute names at will.
- Document structures can be nested to any level of complexity.
- Any XML document can contain an optional description of its grammar for use by applications that need to perform structural validation.
- XML solution is system-independent and vendor-independent.

XML can be read by human due to the tags chosen to represent the data. For example the XML would be written in the tags name "employee" so that when the reader see the XML code, he or she would have the knowledge that they are dealing with employee related information.

8.3.2 Web Wrapper Technology

Professor Madnick of MIT, along with MIT researchers, has developed technology for obtaining the data from the Web [Alter, 1997]. It is called "Web Wrapper". Web Wrapper is a middleware that located in the users' server and has the ability to extract data as if treat Web as a giant database. User can post a SQL query and get back a database record or spreadsheet that contains the information they want. The generation includes a "Web pages spec file" which provides a "schema" (what the database structure should look like), "page transition" (which tells the engine how many pages to go to on a site to satisfy query), and "extraction rules" (guidelines for locating information on Web Pages) [Alter, 1997].

Figure 8-1 shows the steps in solving problems in finding data from the Web.

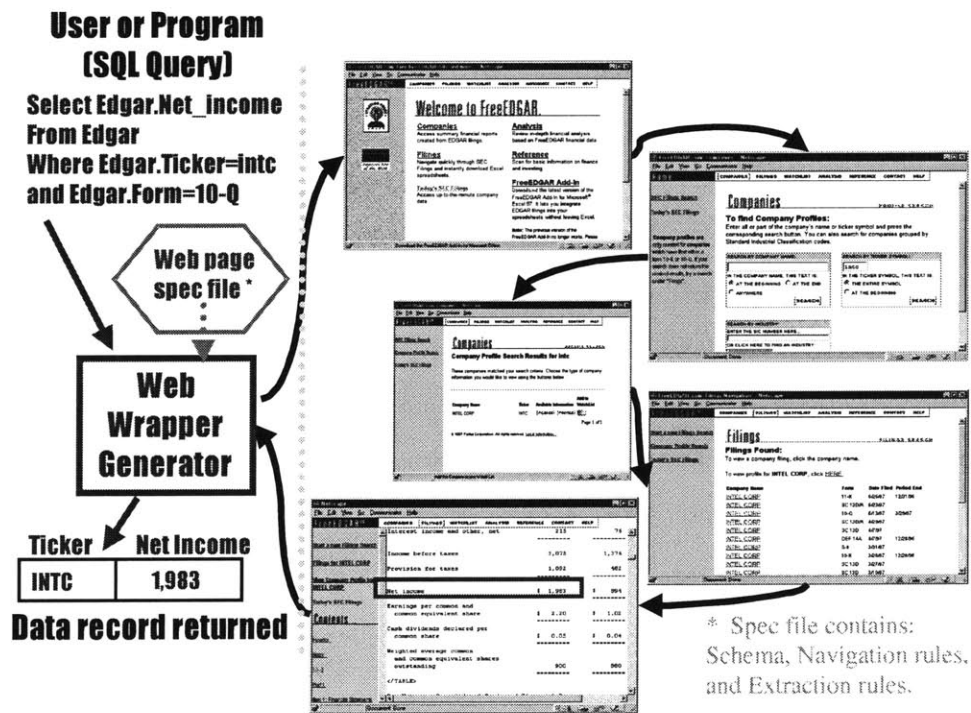


Figure 8-1 Web Wrapper Technology

Source: Course 15.578, Lecture 17 notes, spring 2000

The user has to equip the web wrapper engine with the spec-file. The spec file contains a program that tells the web wrapper engine where to get the data and what to get. The engine then prompts to connect to web server, search the data, and send it to the application or browser.

8.3.3 Content Mediation Engine

The MIT Context Interchange Approach (COIN) project executes a research to interpret the accuracy and data comes from the Web and translate the exact meaning of the query that comes from the Web or database. The architecture of COIN is described in figure 8-3.

Context mediator is an agent that has the ability to ensure the user gets the data that he or she wants.

The Context Interchange Approach

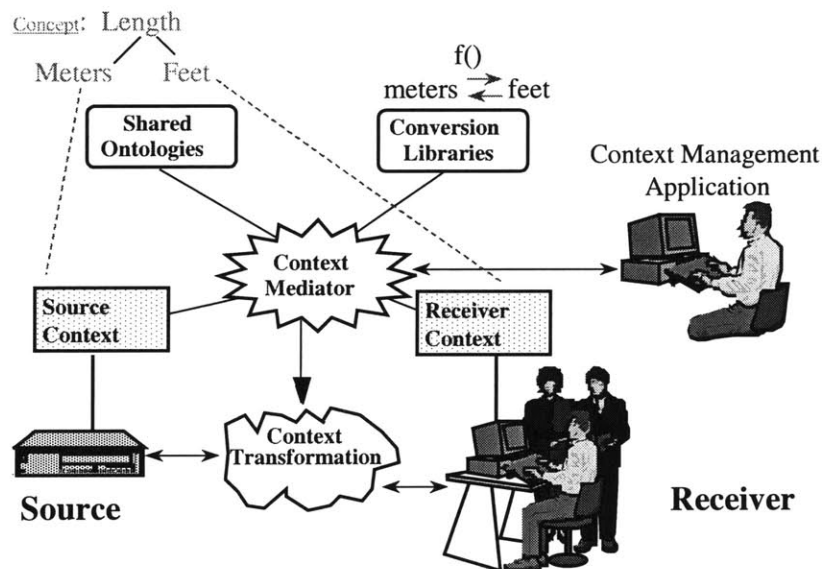


Figure 8-2 Context Mediation Engine

Source: Course 15.578, Lecture 18, Spring 2000

First the user inquires a data using the receiver. The receiver can be browser or other application. However, in the receiver is categorized with type. For example one application is customized for searching in metric. When the data comes in the form of the British unit, then the context mediator can communicate to the conversion libraries to convert the data. After data being converted in conversion library, the data is being sent back to the context mediator and then sent to the user.

8.4 Summary

The primary benefit of the collaboration GUI is that users can read data from distributed sources in one interface. This enables them to save time and increase productivity.

The technology that enables the process of collaboration are XML, web wrapper, and context mediation engine. XML allows the web pages to be structurally composed so that it can be used as a database. The web wrapper enables intelligent data mining to search for information of the web. Finally, the context mediation engine allows data translation according to the user's needs.

The benefit of collaborative GUI is stated in the context below:

Raymond C. Bonker, a vice president at Merrill Lynch, sees a payoff in combatting information overload. His vision: Pull financial data off the Internet, add information from external sources and internal database, and deliver a money-making mix of information to sales staff, researcher and traders in useful, summary form. That could result in better, faster decision-making and less time waste on browsing, calculating or deciphering the many monitors that crowd their desks

Source: Alter Allan, Computer World, September 15, 1997

Figure 8-3 symbolizes the benefit of collaborative display.

Utilizing the collaborative GUI, the user enables to see all the information inside a single interface and have a real-time data updated frequently. With this capability, the collaborative graphical user interface becomes a simple, but informative tools to increase productivity.

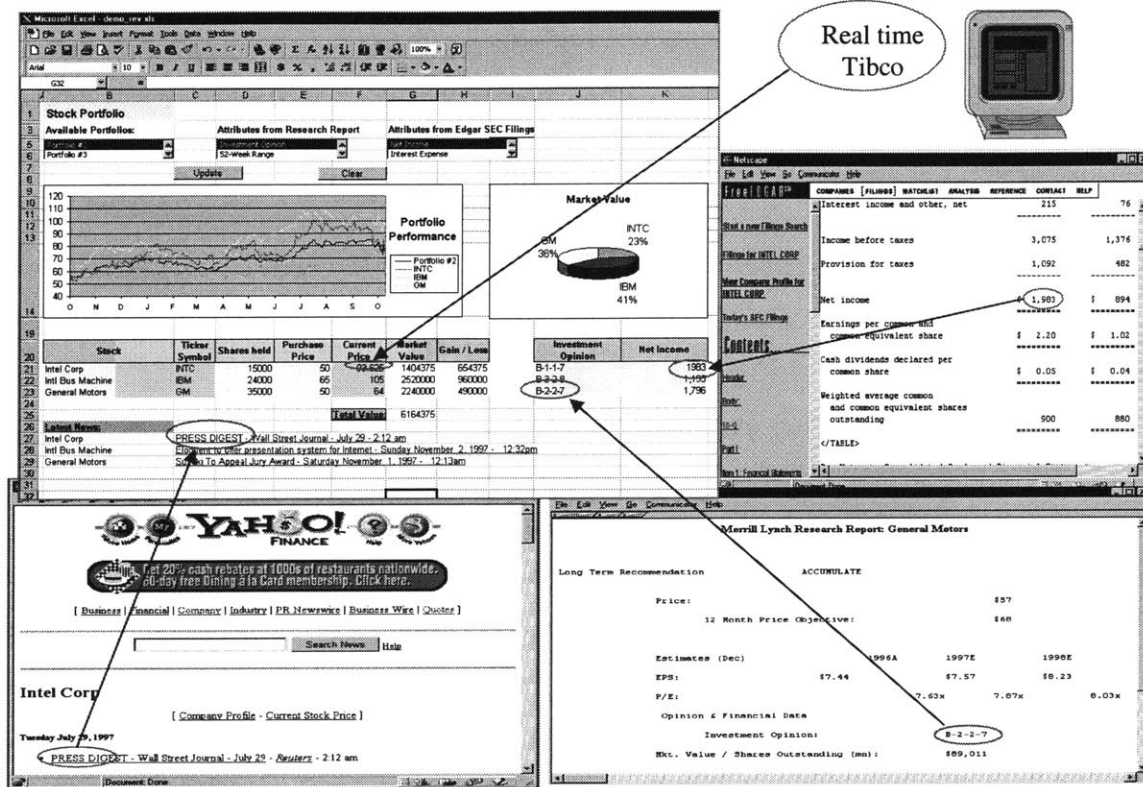


Figure 8-3 Excel Interface as Collaborative Display

Source: Course 15.578, Lecture 17, Spring 2000



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER

MISSING PAGE(S)

Page 94 is missing from the Archives copy.
Best copy available.



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER

MISSING PAGE(S)

Page 95 is missing from the Archives copy.
Best copy available.

References

- Alter, Allan E. *MIT Professor works to create a New Web Order*, www.computerworld.com, September 15, 1997.
- Apple, *Human Interface Guidelines: The Apple Desktop Interface*, Addison-Wesley: Reading, MA, 1987.
- Bosak, Jon. *XML, Java and the Future of the Web*. <http://metalab.unc.edu/pub/sun-info/xml/why/xmlapps.htm>, March 01, 1997.
- Diaper, D. *Task Analysis for Human-Computer Interaction*, Prentice Hall: Englewood Cliffs, NJ, 1989.
- Galitz, Wilbert O. *It's Time to Clean Your Windows*. John Wiley & Sons, Inc: New York, NY, 1997.
- Hix, D. & Hartson, H. R. *Developing User Interface: Ensuring usability through product and process*, John Wiley & Sons, Inc: New York, NY, 1993.
- IBM. *Systems Application Architecture Common User Access*. Addison-Wesley: Reading, MA, 1987.
- IeCollab Project Documentation, collaborate.mit.edu/1.120.html, 2000.
- Madnick, S. *Course 15.565 – Global Information Technology, Lecture 17 & 18*, MIT Sloan, Spring 2000.
- Mayhew, Deborah J. *Principles and Guidelines in Software user Interface Design*. Prentice Hall: Englewood Cliffs, NJ, 1992.
- McConnell, Steve. *Rapid Development*. Microsoft Press: Redmond, WA, 1996
- Microsoft. *The Window Interface: An Application Design Guide*. Microsoft Press, 1992.

- Nielsen, Jakob & del Galdo, Elisa M. *International User Interface*. John Wiley & Sons, Inc: New York, NY, 1996.
- Nielsen, J. *Coordinating User Interface for Consistency*, Academic Press, 1989.
- Olsen, Dan R. *Developing User Interface*. Morgan Kaufmann Publishers, Inc: San Francisco, CA, 1998.
- OSF. *OSF/ Motif Style Guide, revision 1.1*, Prentice Hall: Englewood Cliffs, NY, 1991.
- Peddie, Jon. *Graphical User Interface and Graphic Standard*. McGraw-Hill: New York, NY, 1992.
- Redmond-Pyle, David & Moore, Alan. *Graphical User Interface Design and Evaluation*. Addison-Wesley: Reading, MA, 1992.
- Rumbaugh, J. *Object Oriented Modelling and Design*. Prentice Hall: Englewood Cliffs, NY, 1992.
- Shneiderman, B. *Designing the User Interface*. Addison-Wesley: Reading, MA, 1992.
- Stein, Lincoln D. *How to Set Up and Maintain a Website*. Addison-Wesley: Reading MA, 1997.
- Wagner, A. *The Art of Human-Computer Interaction*, Addison-Wesley: Reading MA, 1992.
- Weinschenk, Susan & Jamar, Pamela. *GUI Design Essential*. John Wiley & Sons, Inc: New York, NY, 1997.
- Wood, Larry E. *User Interface Design*. CRC Press LLC: Boca Raton, FL, 1998.