

# **COURSE GUIDE**

## **CIT 711 USER INTERFACE DESIGN AND ERGONOMICS**

**Course Team**      Dr. A. S. Sodiya (Course Developer)  
University of Agriculture, Abeokuta  
A. A. Afolorunso (Course Coordinator) – NOUN



**NATIONAL OPEN UNIVERSITY OF NIGERIA**

National Open University of Nigeria  
Headquarters  
14/16 Ahmadu Bello Way  
Victoria Island  
Lagos

Abuja Office  
No. 5 Dar es Salaam Street  
Off Aminu Kano Crescent  
Wuse II, Abuja

e-mail: [centralinfo@nou.edu.ng](mailto:centralinfo@nou.edu.ng)

URL: [www.nou.edu.ng](http://www.nou.edu.ng)

National Open University of Nigeria 2012

Printed 2013

ISBN

All Rights Reserved

Printed by .....

For

National Open University of Nigeria

<b>CONTENTS</b>	<b>PAGE</b>
Introduction.....	iv
What you will Learn in this Course.....	iv
Course Aims.....	iv
Course Objectives.....	iv
Working through this Course.....	v
Course Materials.....	v
Study Units .....	vi
Textbooks and References.....	vii
Assignments File .....	vii
Assessment.....	vii
Tutor -Marked Assignments (TMAs) .....	vii
Final Examination and Grading.....	viii
Course Marking Scheme.....	viii
Course Overview.....	ix
How to Get the Most from this Course .....	x
Facilitators/Tutors and Tutorials .....	xi
Summary.....	xii

## **INTRODUCTION**

CIT 711 – User Interface Design and Ergonomics is a three- credit unit course of 20 units. It discusses the introduction, design, implementation and evaluation of user interface. The course material is clearly designed to enhance the understanding of students. The aim of this course is to equip you with the basic skills of studying and understanding the concept of user interface design. The techniques involved will be explicitly covered while evaluation procedures will also be discussed. By the end of the course, you should be able to confidently study, analyse and design a standard user interface.

This course guide gives you a brief overview of the course content, course duration, and course materials.

## **WHAT YOU WILL LEARN IN THIS COURSE**

The main purpose of this course is to provide the necessary tools in analysing and designing user interface systems. It makes available the steps and tools that will enable you to make proper and accurate decision about necessary design technique whenever the need arises. This, we intend to achieve through the following:

### **COURSE AIMS**

- To introduce the concepts user interface design and ergonomics
- To describe the various techniques for designing user interface
- To describe the practical implementation of user interface and
- To discuss the techniques for user interface evaluation.

### **COURSE OBJECTIVES**

Certain objectives have been set out to ensure that the course achieves its aims. Apart from the course objectives, every unit of this course has set objectives. In the course of the study, you will need to confirm at the end of each unit if you have met the objectives set at the beginning of each unit. By the end of this course, you should be able to:

- recognise the essentials of good interface design techniques
- describe in general term, the concept of user interface design
- illustrate and design a standard user interface
- analyse an existing user interface design
- explain the various types of challenges that could be encountered in designing a good user interface
- describe the various process of implementing designs

- discuss how to evaluate a given design.

## **WORKING THROUGH THIS COURSE**

In order to have a thorough understanding of the course units, you will need to read and understand the contents, practice the steps by designing a standard user interface model.

This course is designed to cover approximately 16 weeks, and it will require your devoted attention. You should do the exercises in the Tutor-Marked Assignments and submit to your tutors.

## **COURSE MATERIALS**

These include:

1. Course Guide
2. Study Units
3. Recommended Texts
4. A file for your assignments and for records to monitor your progress.

## **STUDY UNITS**

There are 20 study units in this course:

### **Module 1**

Unit 1	Foundation of User Interface Design
Unit 2	Designing Good User Interfaces
Unit 3	Graphical User Interface (GUI)
Unit 4	Human-Computer Interaction
Unit 5	Ergonomics

### **Module 2**

Unit 1	Human Capabilities in User Interface Design
Unit 2	Understanding Users and Task Analysis
Unit 3	User-Centered Design
Unit 4	Interactive Design
Unit 5	Usability
Unit 6	Interaction Styles and Graphic Design Principles

**Module 3**

Unit 1	Prototyping
Unit 2	Prototyping/Implementation Methods and Toolkits
Unit 3	Input and Output Models
Unit 4	Model- View-Controller (MVC)
Unit 5	Layouts and Constraints

**Module 4**

Unit 1	Techniques for Evaluating and Measuring Interface Usability
Unit 2	Evaluating User Interface without the User
Unit 3	Evaluating User Interface with the User
Unit 4	Other Evaluation Methodologies

Make use of the course materials, do the exercises to enhance your learning.

**TEXTBOOKS AND REFERENCES**

- Dumas, J. S. & Redish, J. C. (1999). *A Practical Guide to Usability Testing* (revised ed.). Bristol, U.K.: Intellect Books.
- Holm, I. (2006). *Ideas and Beliefs in Architecture and Industrial design: How attitudes, orientations, and underlying assumptions shape the built environment*. Oslo School of Architecture and Design. ISBN 8254701741
- Kuniavsky, M. (2003). *Observing the User Experience: A Practitioner's Guide to User Research*, San Francisco, CA: Morgan Kaufmann.
- Mckeown, C. (2008). *Office Ergonomics: Practical Applications*. Boca Raton, FL: Taylor & Francis Group, LLC.
- Molich, R. & Nielsen, J. (1990). Improving a Human-Computer Dialogue: What Designers know about Traditional Interface Design." *Communications of the ACM*, 33, pp. 338-342.
- Nielsen, J. & Molich, R. (1990). "Heuristic Evaluation of User Interfaces." Proc. CHI'90 Conference on Human Factors in Computer Systems. New York: ACM, pp. 249-256.
- Nielsen, J. (1992). "Finding Usability Problems through Heuristic Evaluation." Proc. CHI'92 Conference on Human Factors in Computer Systems. New York: ACM, pp. 373-380.

Nielsen, J. (1992). *“Usability Engineering.”* San Diego, CA: Academic Press.

Wharton, C., Rieman, J., Lewis, C. & Polson, P. (1994). “The Cognitive Walkthrough: A Practitioner’s Guide.” In Nielsen, J. & Mack, R. L. (Eds.), *Usability Inspection Methods*, New York: John Wiley & Sons.

Wickens, C. D *et al.* (2004). *An Introduction to Human Factors Engineering* (2nd ed.). Pearson Education, Inc., Upper Saddle River, NJ: Prentice Hall.

[www.wikipedia.org](http://www.wikipedia.org).

## **ASSIGNMENTS FILE**

Tutor-Marked Assignment is a supervised assignment. The assignments take a certain percentage of your total score in this course. The Tutor-Marked Assignments will be assessed by your tutor within a specified period. The examination at the end of this course will aim at determining the level of mastery of the subject matter. This course includes 16 Tutor-Marked Assignments and each must be done and submitted accordingly. Your best scores however, will be recorded for you. Be sure to send these assignments to your tutor before the deadline to avoid loss of marks.

## **ASSESSMENT**

There are two aspects to the assessment of the course. First are the Tutor- Marked Assignments; second, is a written examination.

In tackling the assignments, you are expected to apply information and knowledge acquired during this course. The assignments must be submitted to your tutor for formal assessment in accordance with the deadlines stated in the assignment file. The work you submit to your tutor for assessment will count for 30% of your total course mark.

At the end of the course, you will need to sit for a final three-hour examination. This will also count for 70% of your total course mark.

## **TUTOR MARKED ASSIGNMENTS (TMAS)**

There are 16 Tutor- Marked Assignments in this course. You need to submit all the assignments, the total marks for the best four assignments will be 30% of your total course mark.

Assignment questions for the units in this course are contained in the assignment file. You should be able to complete your assignments from the information and materials contained in your set textbooks, reading and studying units. However, you may wish to use other references to broaden your viewpoint and provide a deeper understanding of the subject.

When you have completed each assignment send it to your tutor. Make sure that each assignment reaches your tutor on or before the deadline given. If, however, you cannot complete your work on time, contact your tutor before the assignment is done to discuss the possibility of an extension.

## **FINAL EXAMINATION AND GRADING**

The final examination for the course will carry 70% of the total marks available for this course. The examination will cover every aspect of the course, so you are advised to revise all your corrected assignments before the examination.

This course endows you with the status of a teacher and that of a learner. This means that you teach yourself and that you learn, as your learning capabilities would allow. It also means that you are in a better position to determine and to ascertain the what, the how, and the when.

The course units are similarly designed with the introduction following the table of contents, then a set of objectives and then the dialogue and so on.

The objectives guide you as you go through the units to ascertain your knowledge of the required terms and expressions.

## **COURSE MARKING SCHEME**

This table shows how the actual course marking is broken down.

<b>Assessment</b>	<b>Marks</b>
Assignment 1- 16	16 assignments, best six count for 30% of course marks
Final Examination	70% of overall course marks
Total	100% of course marks



## COURSE OVERVIEW

Unit	Title of Work	Weeks Activity	Assessment (End of Unit)
	Course Guide	Week 1	
	<b>MODULE 1</b>		
1,2	Fundamentals of User Interface Design and Designing Good User Interfaces	Week 1	Assignment 1
3	Graphical User Interface (GUI)	Week 2	Assignment 2
4	Human-Computer Interaction	Week 3	Assignment 3
5	Ergonomics	Week 4	Assignment 4
	<b>MODULE 2</b>		
1	Human Capabilities in User Interface Design	Week 5	Assignment 5
2	Understanding Users and Task Analysis	Week 6	Assignment 6
3	User-Centered Design	Week 7	Assignment 7
4,5	Usability, Interaction Styles and Graphic Design Principles	Week 8	Assignment 8
	<b>MODULE 3</b>		
1,2	Prototyping and Implementation Methods and Toolkits	Week 9	Assignment 9
3	Input and Output Models	Week 10	Assignment 10
4	Model -View-Controller (MVC)	Week 11	Assignment 11
5	Layouts and Constraints	Week 12	Assignment 12
	<b>MODULE 4</b>		
1	Techniques for Evaluating and Measuring Interface Usability	Week 13	Assignment 13
2	Evaluating User Interface without the User	Week 14	Assignment 14
3	Evaluating User Interface with the Users	Week 15	Assignment 15
4	Other Evaluation Methodologies	Week 16	Assignment 16
	Revision	Week 17	
	Examination	Week 18	
Total		18 weeks	

## HOW TO GET THE MOST FROM THIS COURSE

In distance learning, the study units replace the university lecturer. This is one of the great advantages of distance learning; you can read and work through specially designed study materials at your own pace, and at a time and place that suit you best. Think of it as reading the lecture instead of listening to a lecturer. In the same way that a lecturer might set you some reading to do, the study units tell you when to read your set books or other material. Just as a lecturer might give you an in-class exercise, your study units provide exercises for you to do at appropriate points.

Each of the study units follows a common format. The first item is an introduction to the subject matter of the unit and how a particular unit is integrated with the other units and the course as a whole. Next is a set of learning objectives. These objectives enable you know what you should be able to do by the time you have completed the unit. You should use these objectives to guide your study. When you have finished the units you must go back and check whether you have achieved the objectives. If you make a habit of doing this you will significantly improve your chances of passing the course.

Remember that your tutor's job is to assist you. When you need help, don't hesitate to call and ask your tutor to provide it.

1. Read this course guide thoroughly.
2. Organise a study schedule. Refer to the 'course overview' for more details. Note the time you are expected to spend on each unit and how the assignments relate to the units. Whatever method you chose to use, you should decide on it and write in your own dates for working on each unit.
3. Once you have created your own study schedule, do everything you can to stick to it. The major reason that students fail is that they lag behind in their course work.
4. Turn to unit 1 and read the introduction and the objectives for the unit.
5. Assemble the study materials. Information about what you need for a unit is given in the 'overview' at the beginning of each unit. You will almost always need both the study unit you are working on and one of your set of books on your desk at the same time.

6. Work through the unit. The content of the unit itself has been arranged to provide a sequence for you to follow. As you work through the unit you will be instructed to read sections from your set books or other articles. Use the unit to guide your reading.
7. Review the objectives for each study unit to confirm that you have achieved them. If you feel unsure about any of the objectives, review the study material or consult your tutor.
8. When you are confident that you have achieved a unit's objectives, you can then start on the next unit. Proceed unit by unit through the course and try to pace your study so that you keep yourself on schedule.
9. When you have submitted an assignment to your tutor for marking, do not wait for its return before starting on the next unit. Keep to your schedule. When the assignment is returned, pay particular attention to your tutor's comments, both on the tutor-marked assignment form and also written on the assignment. Consult your tutor as soon as possible if you have any questions or problems.
10. After completing the last unit, review the course and prepare yourself for the final examination. Check that you have achieved the unit objectives (listed at the beginning of each unit) and the course objectives (listed in this course guide).

## **FACILITATORS/TUTORS AND TUTORIALS**

There are 15 hours of tutorials provided in support of this course. You will be notified of the dates, times and location of these tutorials, together with the name and phone number of your tutor, as soon as you are allocated a tutorial group.

Your tutor will mark and comment on your assignments, keep a close watch on your progress and on any difficulties you might encounter and provide assistance to you during the course. You must mail or submit your Tutor-Marked Assignments to your tutor well before the due date (at least two working days are required). They will be marked by your tutor and returned to you as soon as possible.

Do not hesitate to contact your tutor by telephone, or e-mail if you need help. The following might be circumstances in which you would find help necessary. Contact your tutor if:

- you do not understand any part of the study units or the assigned readings
- you have difficulty with the self-tests or exercises
- you have a question or problem with an assignment, with your tutor's comments on an assignment or with the grading of an assignment.

You should try your best to attend the tutorials. This is the only chance to have face to face contact with your tutor and to ask questions which are answered instantly. You can raise any problem encountered in the course of your study. To gain the maximum benefit from course tutorials, prepare a question list before attending them. You will learn a lot from participating in discussions actively.

## **SUMMARY**

User Interface Design and Ergonomics deals with the analysis, design, implementation and evaluation of user interface design. The aim of this course is to equip you with the basic skills of studying and understanding the concept behind user interface design. By the end of the course, you will be able to confidently study, analyse and design a standard user interface. The content of the course material was planned and written to ensure that you acquire the proper knowledge and skills for the appropriate situations. Real-life situations have been created to enable you identify with and create some of your own. The essence is to get you to acquire the necessary tools, we hope to achieve that.

I wish you success with the course and hope that you will find it both interesting and useful.

**MAIN  
COURSE**

<b>CONTENTS</b>		<b>PAGE</b>
<b>Module 1</b>	.....	<b>1</b>
Unit 1	Foundation of User Interface Design.....	1
Unit 2	Designing Good User Interfaces.....	8
Unit 3	Graphical User Interface (GUI).....	18
Unit 4	Human-Computer Interaction.....	28
Unit 5	Ergonomics.....	35
<b>Module 2</b>	.....	<b>45</b>
Unit 1	Human Capabilities in User Interface Design	45
Unit 2	Understanding Users and Task Analysis.....	58
Unit 3	User-Centered Design.....	69
Unit 5	Usability.....	77
Unit 6	Interaction Styles and Graphic Design Principles.....	86
<b>Module 3</b>	.....	<b>97</b>
Unit 1	Prototyping.....	97
Unit 2	Prototyping/Implementation Methods and Toolkits.....	104
Unit 3	Input and Output Models.....	120
Unit 4	Model- View-Controller (MVC).....	131
Unit 5	Layouts and Constraints.....	139
<b>Module 4</b>	.....	<b>143</b>
Unit 1	Techniques for Evaluating and Measuring Interface Usability.....	143
Unit 2	Evaluating User Interface without the User	153
Unit 3	Evaluating User Interface with the User.....	168
Unit 4	Other Evaluation Methodologies.....	188

## **MODULE 1 INTRODUCTION TO USER INTERFACE DESIGN**

Unit 1	Fundamentals of User Interface Design
Unit 2	Designing Good User Interfaces
Unit 3	Graphical User Interface (GUI)
Unit 4	Human-Computer Interaction
Unit 5	Ergonomics

### **UNIT 1 FUNDAMENTALS OF USER INTERFACE DESIGN**

#### **CONTENTS**

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	User Interface
3.2	Significance of User Interface
3.3	Types of User Interfaces
3.4	History of User Interfaces
3.5	User Interface Modes and Modalities
3.6	Introduction to Usability
4.0	Conclusion
5.0	Summary
6.0	Tutor- Marked Assignment
7.0	References/Further Reading

#### **1.0 INTRODUCTION**

Having read through the course guide, you will have a general understanding of what this unit is about and how it fits into the course as a whole. This unit describes the general fundamentals of user interface design.

#### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain the term user interface design
- identify the significance of user interface
- explain the history behind user interfaces
- describe the modalities and modes of user interface.

### 3.0 MAIN CONTENT

#### 3.1 User Interface

The **user interface** (also known as Human Machine Interface (HMI) or Man-Machine Interface (MMI)) is the aggregate of means by which people—the *users*—interact with the *system*—a particular machine, device, computer program or other complex tool.

**User interface** is the point at which a user or a user department or organisation interacts with a computer system. The part of an interactive computer program that sends messages to and receives instructions from a terminal user.

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

#### **User Interface Design or User Interface Engineering**

This is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centred design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design may be utilised to apply a theme or style to the interface without compromising its usability. The design process must balance the technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialise in certain types of projects and have skills centered around their expertise, e.g., software design, user research, web design, or industrial design.

## 3.2 Significance of User Interface

To work with a system, users have to be able to control the system and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windscreen and the exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

The term *user interface* is often used in the context of computer systems and electronic devices. The user interface of a mechanical system, a vehicle or an industrial installation is sometimes referred to as the **Human-Machine Interface (HMI)**. HMI is a modification of the original term MMI (Man-Machine Interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, which is more commonly used for Human-Computer *Interaction* than Human-Computer *Interface*. Other terms used are Operator Interface Console (OIC) and Operator Interface Terminal (OIT).

## 3.3 Types of User Interfaces

Currently (as at 2009) the following types of user interface are the most common:

- **Graphical User Interfaces (GUI)** accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-Oriented User Interfaces (OOUIs) and application oriented interfaces. Examples are Windows Operating System, LabView, etc.
- **Web-based User Interfaces or Web User Interfaces (WUI)** accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilise Java, AJAX, Adobe Flex, Microsoft, NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called control panels.



User interfaces that are common in various fields outside desktop computing:

- **Command Line Interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Examples are MS-DOS and UNIX interface. This interface is also used for system administration tasks, etc.
- **Tactile Interfaces** supplement or replace other forms of output with haptic feedback methods. This is used in automated simulators and so on.
- **Touch User Interface** are graphical user interfaces using a touchscreen display as a combined input and output device. It is used in many types of point of sale, industrial processes and machines, self-service machines etc. Examples are touch screen monitors.

Other types of user interfaces are:

- **Attentive User Interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch Interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based Interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture Interface** are graphical user interfaces which accept input in form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent User Interfaces** are human-machine interfaces that aim at improving the efficiency, effectiveness, and naturalness of

human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).

- **Motion Tracking Interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple.
- **Multi-screen Interfaces** employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Non-command User Interfaces** allow users to infer their needs and intentions, without requiring explicit commands formulation.
- **Object-Oriented User Interface (OOUI)** examples of OOUI are:

**Reflexive User Interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.

**Tangible User Interfaces** which place a greater emphasis on touch and physical environment or its element.

**Text User Interfaces** are user interfaces which output text, but accept other forms of input in addition to or in place of typed command strings.

**Voice User Interfaces** which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.

**Natural-Language Interfaces** used for search engines and on webpages. User types in a question and waits for a response.

**Zero-Input Interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.

**Zooming User Interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail

## SELF-ASSESSMENT EXERCISE

Check the GUI features and functions of LabView. Then compare their environments with MS-DOS.

### 3.4 History of User Interfaces

The history of user interfaces can be divided into the following phases according to the dominant type of user interface:

- Batch Interface, 1945-1968
- Command-line User Interface, 1969 to present
- Graphical User Interface, 1981 to present

### 3.5 User Interface Modalities and Modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

Input — allowing the users to manipulate a system. For example the computer keyboard allows the user to enter typed text; digitising tablet allows the user to create free-form drawing.

Output — allowing the system to indicate the effects of the users' manipulation. For example the computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*).

The user interface may employ several redundant input and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending on the state of the computer program. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

## 4.0 CONCLUSION

In this unit, you have been introduced to the fundamental concepts of user interface. You have also learnt the history and significance of user interface design.

## 5.0 SUMMARY

In this unit, you have learnt the:

- introduction of user interface which is the aggregate of means by which users interact with a particular machine, device, computer program or any other complex tool
- study of the various types of user interface design which includes graphical user interfaces, web-based user interfaces, command line interfaces e.t.c.
- history of user interfaces which can be divided into batch interface, command-line user interface and graphical user interface
- modality of a user interface which is a path of communication employed by the user interface to carry input and output.

## 6.0 TUTOR-MARKED ASSIGNMENT

- i. Explain the Microsoft's Clippy the paperclip.
- ii. Write a short note on the Command-line User Interface.

## 7.0 REFERENCES/FURTHER READING

Pinel, J. P. (2008). *Biopsychology* (7th ed.). Boston: Pearson. p. 357.  
Wikipedia.org

## UNIT 2 DESIGNING GOOD USER INTERFACES

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Essentials of Interface Design
    - 3.1.1 Designing a Good Interface
    - 3.1.2 Tips for Designing Good User Interface
  - 3.2 Understanding Users
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the essentials of designing good interface designs and also discusses the various users.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the essentials of a good interface design
- identify the necessary tips needed for designing a good interface
- discuss various users.

### 3.0 MAIN CONTENT

#### 3.1 Essentials of Interface Design

There are three pillars to an application's success:

- Features
- Function
- Face

*Features* refer to what the application will do for the user. Features are the requirements for the software.

*Function* refers to how well the software operates. Bug-free software will function perfectly.

**Face** refers to how the application presents itself to the user; the application's "user interface."

Features, function and face can be restated as questions:

- i. Does the software meet the user's requirements? (Features)
- ii. Does the software operate as intended? (Function)
- iii. Is the software easy to use? (Face)

The goal of user interface design is to put a happy face on the application. Stated in more concrete terms, a successful user interface will require *Zero Training* and will be *friendly not foreboding*.

### ***Zero Training***

The goal of Zero Training could be considered as a sub-goal of friendly not foreboding. However, training costs are a major impediment to the usage of software making Zero Training an important goal by itself.

There are two types of training involved in software design: software training and job training. Software training assumes the user knows how to do the job at hand and only needs to learn how to use the software to do the job. Job training teaches the user how to do the job - which can include more than how to use the application to do the job. The goal of Zero Training relates to *zero software training*. Job training can be integrated with software training, but results in a much more ambitious project.

### ***Friendly not Foreboding***

Almost everything you do to implement the goal of Zero Training will further the goal of being friendly not foreboding. However, some techniques for reducing training may slow up experienced users. For example, you could pop-up new user messages whenever the user lands in a particular field. Seeing the same message after awhile makes the experienced user dispense with the messages.

Being friendly is an attitude and encompasses more than what is necessary for the Zero Training goal. Applications do have an attitude. For example, consider the following sets of application messages:

"Database does not exist" "I could not find database "CorpInfo".  
If you are sure this name is correct,  
CorpInfo could be unavailable due to  
maintenance or LAN problems. You

should contact the help desk to see when CorpInfo will again be available.”

“SQL access error 123” “I could not save information to the database. You can try to save again to see if the error clears. If you call the help desk concerning this problem, tell them you have a “SQL access error 123”.

“out of hunk”<sup>1</sup> “I have run out of memory (RAM). This typically happens when there is a bug in the program which causes it to lose memory over time. Save your game if possible. To free the memory you will need to reset the computer (turn it off and then on).”

The attitude of the first message is “you did something that caused me a problem” while the attitude of the second message is “I have a problem. Could you give me a hand?”

### 3.1.1 Designing a Good User Interface

Designing a good user interface is an iterative process. First, you design and implement a user interface using appropriate techniques. Then you evaluate the design. The results of the evaluation feed the next design and implementation. You stop the process when you have met your design goals or you run out of time and/or money.

Note that if you have different user communities (or the same user with different jobs), you may need different user interfaces, customisable user interfaces or both. For example, Microsoft Word provides four user interfaces: normal, outline, page layout and master. In addition, Microsoft Word provides a host of customisation features for the keyboard, menu and toolbars.

While design is important, the real key to creating a good user interface is in your evaluation techniques. Obviously, you should use your own user interface. If you can't use it, how can anyone else? Next, get feedback from your testers.

The best evaluations are done by watching over the shoulder of the user. The key here is watching. If you are telling the user what to do, you will never find out if your interface is easy to use. Let the user figure it out himself or herself. If the user has to ask you what to do or pauses to figure out what to do next, you may need to work on your interface. If

the user grimaces, find out why. Learn from the experience. Some of the most innovative designs were shot down when the users could not figure them out.

You will need both new and experienced users for testing your interface. The new users will help you determine if you meet the Zero Training goal. The experienced users will let you know if your methods for meeting the Zero Training goal interfere with getting work done once the user has learned the software.

### 3.1.2 Tips for Designing a Good User Interface

- **Consistency:** The most important thing that you can possibly do is make sure that your user interface works consistently. If you can double-click on items in one list and have something happen, then you should be able to double-click on items in any other list and have the same sort of thing happen. Put your buttons in consistent places on all of your windows, use the same wording in labels and messages, and use a consistent colour scheme throughout. Consistency in your user interface allows your users to build an accurate mental model of the way that it works, and accurate mental models lead to lower training and support costs.
- **Set standards and stick to them:** The only way that you'll be able to ensure consistency within your application is to set design standards and then stick to them. The best approach is to adopt an industry standard and then fill any missing guidelines that are specific to your needs. Industry standards, such as the ones set by IBM (1993) and Microsoft (1995), will often define 95-99% of what you need. By adopting industry standards, you are not only taking advantage of the work of others, you are also increasing the chances that your application will look and feel like other applications that your users purchase or have built. User interface design standards should be set during the define infrastructure stage.
- **Explain the rules:** Your users need to know how to work with the application that you built for them. When an application works consistently, it means you only have to explain the rules once. This is a lot easier than explaining in detail exactly how to use each and every feature in an application step by step.
- **Support both novices and experts:** Although a library-catalog metaphor might be appropriate for casual users of a library system, library patrons, it probably is not all that effective for expert users, librarians. Librarians are highly trained people who



are able to use complex search systems to find information in a library; therefore you should consider building a set of search screens to support their unique needs.

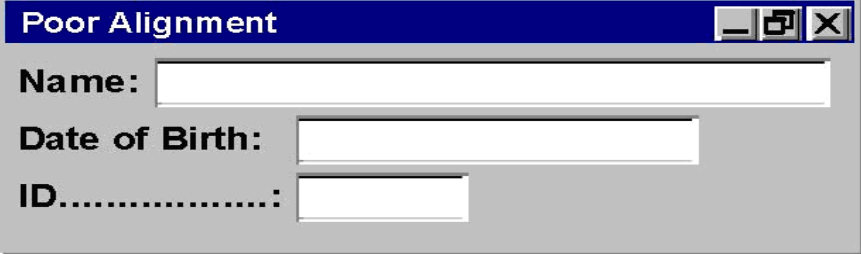
- **Navigation between screens is important:** If it is difficult to get from one screen to another, then your users will quickly become frustrated and give up. When the flow between screens matches the flow of the work that the user is trying to accomplish, then your application will make sense to your users. Because different users work in different ways, your system will need to be flexible enough to support their various approaches. Interface-flow diagrams can be used during the model stage to model the flow between screens.
- **Navigation within a screen is important:** In Western societies, people read left to right and top to bottom. Because people are used to this, so you should design screens that are also organised left to right and top to bottom. All you want is to organise navigation between widgets on your screen in a manner that users will find familiar to them.
- **Word your messages and labels appropriately:** The text that you display on your screens is a primary source of information for your users. If your text is worded poorly then your interface will be perceived poorly by your users. Using full words and sentences, as opposed to abbreviations and codes makes your text easier to understand. Your messages should be worded positively, imply that the user is in control, and provide insight into how to use the application properly. For example, which message do you find more appealing “You have input the wrong information” or “An account number should be 8 digits in length?” Furthermore, your messages should be worded consistently and displayed in a consistent place on the screen. Although the messages “The person’s first name must be input.” and “An account number should be input.” are separately worded well, together they are inconsistent. In light of the first message, a better wording of the second message would be “The account number must be input” to make the two messages consistent.
- **Understand your widgets:** You should use the right widget (widgets are interface elements that the users interact with) for the right task, helping to increase the consistency in your application and probably making it easier to build the application in the first place. The only way that you can learn how to use widgets properly is to read and understand the user-interface standards and guidelines that your organisation has adopted.

- **Look at other applications with a grain of salt:** Unless you know that another application follows the user-interface standards and guidelines of your organisation, you must not assume that the application is doing things right. Although it is always a good idea to look at the work of others to get ideas, until you know how to distinguish between good and bad user-interface design you have to be careful. Too many developers make the mistake of imitating the user interface of another application that was poorly designed.
- **Use colour appropriately:** Colour should be used sparingly in your applications, and if you do use it you must also use a secondary indicator. The problem is that some of your users may be colour blind – if you are using colour to highlight something on a screen, then you need to do something else to make it stand out if you want people to notice it, such as display a symbol beside it. You also want to use colours in your application consistently so that you have a common look and feel throughout your application. Also, colour generally does not port well between platforms – what looks good on one system often looks poor on another system. We have all been to presentations where the presenter said “it looks good on my machine at home.”
- **Follow the contrast rule:** If you are going to use colour in your application, you need to ensure that your screens are still readable. The best way to do this is to follow the contrast rule: Use dark text on light backgrounds and light text on dark backgrounds. It is very easy to read blue text on a white background but very difficult to read blue text on a red background. The problem is that there is not enough contrast between blue and red to make it easy to read, whereas there is a lot of contrast between blue and white.
- **Use fonts appropriately:** Old English fonts might look good on the covers of William Shakespeare’s plays, but they are really difficult to read on a screen. Use fonts that are easy to read, such as serif fonts, Times Roman. Furthermore, use your fonts consistently and sparingly. A screen using two or three fonts effectively looks a lot better than a screen that uses five or six. Never forget that you are using a different font every time you change the size, style (bold, italics, underlining,), typeface, or colour.
- **Grey things out, do not remove them:** You often find that at certain times it is not applicable to give your users access to all the functionality of an application. You need to select an object

before you can delete it, so to reinforce your mental model the application should do something with the Delete button and/or menu item. Should the button be removed or greyed out? Grey it out, never remove it. By greying things out when they shouldn't be, users can start building an accurate mental model as to how your application works. If you simply remove a widget or menu item instead of greying it out then it is much more difficult for your users to build an accurate mental model, because they only know what is currently available and not what is not available to them. The old adage that out of sight is out of mind is directly applicable here.

- **Use non destructive default buttons:** It is quite common to define a default button on every screen, the button that gets invoked if the user presses the return/enter key. The problem is that sometimes people will accidentally hit the enter/return key when they do not mean to, consequently invoking the default button. Your default button shouldn't be something that is potentially destructive, such as delete or save (perhaps your user really did not want to save the object at that moment).
- **Alignment of fields:** When a screen has more than one editing field, you need to organise the fields in a way that is both visually appealing and efficient. As shown in figure 1.I, the best way to do so is to left-justify edit fields, or in other words make the left-hand side of each edit field line up in a straight line, one over the other. The corresponding labels should be right justified and placed immediately beside the field. This is a clean and efficient way to organise the fields on a screen.
- **Justify data appropriately:** For columns of data it is common practice to right justify integers, decimal align floating point numbers, and left justify strings.
- **Do not create busy screens:** Crowded screens are difficult to understand and hence are difficult to use. Experimental results show that the overall density of the screen should not exceed 40%, whereas local density within groupings shouldn't exceed 62%.
- **Group things on the screen effectively:** Items that are logically connected should be grouped together on the screen to communicate that they are connected, whereas items that have nothing to do with each other should be separated. You can use whitespace between collections of items to group them and/or you can put boxes around them to accomplish the same thing.

- **Open windows in the centre of the action:** When your user double-clicks on an object to display its edit/detail screen then his or her attention is on that spot. Therefore it makes sense to open the window in that spot, not somewhere else.
- **Pop-up menus should not be the only source of functionality:** Your users cannot learn how to use your application if you hide major functionality from them. One of the most frustrating practices of developers is to misuse pop-up, also called context-sensitive, menus. Typically there is a way to use the mouse on your computer to display a hidden pop-up menu that provides access to functionality that is specific to the area of the screen that you are currently working on.




Poor Alignment

Name:

Date of Birth:

ID.....:



Good Alignment

Name:

Date of Birth:

ID:

Fig. 1.1: Showing that alignment of fields is critical

### 3.2 Understanding Users

You must understand the user to be able to put a happy face on your application. You should understand the user's job, how the software fits in with that job and how the user goes about getting the job done. You need to approach the design of software from the user's viewpoint not from an abstract requirements document. Specifically, you should understand what the user will be doing with the application. If you can think like a user, you can create a much better user interface.

Here are some basic principles to remember about users:

- Your software is like a hammer - The user doesn't really care how well crafted it is, the user just wants nails put in the wall. Users just want to do their job (or play their game). They don't

care about you or your software. Your software is just an expedient tool to take the user where the user wants to go.

- Given a selection of hammers to buy at the hardware store - The user will select the one which will be most fun to use. Of course, this varies by user - some will want the plastic handle, some the wood, some the green, etc. When evaluating your software, users are often swayed by looks, not function. Thus, steps taken to make the product look good (nice icons, pictures, good colour scheme, fields aligned, etc.) will often favourably enhance evaluations of your software.
- It had better drive nails - The user will not really know if your software is adequate to the job until the user has used the software to do the actual work. From an interface perspective, the software should not look like it can do more than it can.
- Some users will try to use a hammer to drive a screw - If your software is good, some user somewhere will try to use the software for some purpose for which you never intended it to be used. Obviously, you cannot design a user interface to deal with uses you cannot foresee. There is no single rigid model of the right way to use the software, so build in flexibility.
- Users will not read an instruction manual for a hammer- They won't read one for your software either, unless they really have to. Users find reading instruction manuals almost as unpleasurable as dental work.
- A user reading the instruction manual for a hammer is in serious trouble - When you create your help system (and printed manual), remember that the user will only resort to those materials if he or she is in trouble. The user will want a problem solved as fast and as easily as possible.
- Hammers don't complain - You should try to eliminate error messages and any error messages your program needs should have the right attitude.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the essentials of good interface design. You have also learnt the necessary tips needed for designing a good interface and the need for understanding various users.

## 5.0 SUMMARY

In this unit, we have learnt:

- the essentials of interface design with emphasis on the features, functions and the face of the software
- that designing a good user interface which has been described as an iterative process involves designing, implementing, evaluating and redesigning until all removable errors have been taken care of
- the tips necessary for designing a good user interface which includes consistency, setting standards and sticking to them, supporting of both novices and experts, e.t.c.
- understanding the user's job, how the software fits in with that job and how the user goes about getting the job done.

### SELF- ASSESSMENT EXERCISE

- i. How do you ensure that the interface supports both novices and experts?
- ii. Write short notes on any three tips necessary for designing a good user interface.

## 6.0 TUTOR- MARKED ASSIGNMENT

- i. How do you ensure that the interface support both novices and expert?
- ii. Write short note on the design of a user interface for a user with hearing disability.

## 7.0 REFERENCES/FURTHER READING

Wikipedia.org

[www://www.linfo.org/gui.html](http://www.linfo.org/gui.html)

## UNIT 3 GRAPHICAL USER INTERFACE

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Graphical User Interface
  - 3.2 History of Graphical User Interface
  - 3.3 Elements of Graphical User Interface
  - 3.4 Three Dimensional Graphical User Interface
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the general concept of Graphical User Interface (GUI) and also the history and elements of graphical user interface. The concept of three dimensional (3D) graphical user interface is also introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe a graphical user interface
- explain the history behind graphical user interface
- list the elements of a graphical user interface
- describe the three-dimensional user interfaces.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Graphical User Interface

Graphical User Interfaces, also known as GUIs, offer a consistent visual language to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. This explains the most common elements of the visual language interfaces.

A **graphical user interface** is a type of user interface which allows people to interact with electronic devices such as computers; hand-held devices such as MP3 Players, Portable Media Players or Gaming

devices; household appliances and office equipment with images rather than text commands. A *GUI* offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

The term *GUI* is historically restricted to the scope of two-dimensional display screens with display resolutions capable of describing generic information, in the tradition of the computer science research at Palo Alto Research Centre (PARC). The term *GUI* earlier might have been applicable to other high-resolution types of interfaces that are non-generic, such as videogames, or not restricted to flat screens, like volumetric displays.

## 3.2 History of Graphical User Interface

### Precursor to GUI

The precursor to GUIs was invented by researchers at the Stanford Research Institute, led by Douglas Engelbart. They developed the use of text-based hyperlinks manipulated with a mouse for the On-Line System. The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC, who went beyond text-based hyperlinks and used a GUI as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. As a result, some people call this class of interface PARC User Interface (PUI) (note that PUI is also an acronym for Perceptual User Interface). Ivan Sutherland developed a pointer-based system called the Sketchpad in 1963. It used a light-pen to guide the creation and manipulation of objects in engineering drawings.

### PARC User Interface

The PARC User Interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC User Interface employs a pointing device in addition to a keyboard. These aspects can be emphasised by using the alternative acronym WIMP, which stands for **W**indows, **I**cons, **M**enus and **P**ointing device.

### Evolution

Following PARC, the first GUI-centric computer operating model was the Xerox 8010 Star Information System in 1981 followed by the Apple Lisa (which presented concept of menu bar as well as window controls) in 1982 and the Atari ST and Commodore Amiga in 1985.



The GUIs that are familiar to most people today are Microsoft Windows, Finder Interface (Mac OS X), and the X Window System interfaces. Apple, IBM and Microsoft used many of Xerox's ideas to develop products, and IBM's Common User Access specifications formed the basis of the user interface found in Microsoft Windows, IBM OS/2 Presentation Manager, and the Unix Motif toolkit and window manager. These ideas evolved to create the interface found in current versions of Microsoft Windows, as well as in Mac OS X and various desktop environments for Unix-like operating systems, such as Linux. Thus most current GUIs have largely common idioms.

### **Post-WIMP Interfaces**

Smaller mobile devices such as PDAs and smart phones typically use the WIMP elements with different unifying metaphors, due to constraints in space and available input devices. Applications for which WIMP is not well suited may use newer interaction techniques, collectively named as post-WIMP user interfaces.

Some touch-screen-based operating systems such as Apple's iPhone OS currently use post-WIMP styles of interaction. The iPhone's use of more than one finger in contact with the screen allows actions such as pinching and rotating, which are not supported by a single pointer and mouse.

A class of GUIs sometimes referred to as post-WIMP include 3D compositing window manager such as Compiz, Desktop Window Manager, and LG3D. Some post-WIMP interfaces may be better suited for applications which model immersive 3D environments, such as Google Earth.

## **3.3 Elements of Graphical User Interfaces**

A GUI uses a combination of technologies and devices to provide a platform which the user can interact with in order to achieve the tasks of gathering and producing information.

Series of elements conforming to visual languages have evolved to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. The most common combination of such elements in GUIs is the WIMP paradigm, especially in personal computers.

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of

the static elements on which the user can interact, and define the appearance of the interface.

The key elements of GUI are divided into two categories viz Structural and Interactive elements.

### 3.3.1 Structural Elements

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of the static elements on which the user can interact, and define the appearance of the interface.

#### Window

A window is an area on the screen that displays information, with its contents being displayed independently from the rest of the screen. An example of a window is what appears on the screen when the "My Documents" icon is clicked in the Windows Operating System. It is easy for a user to manipulate a window: it can be opened and closed by clicking on an icon or application, and it can be moved to any area by dragging it (that is, by clicking in a certain area of the window – usually the title bar along the top – and keeping the pointing device's button pressed, then moving the pointing device). A window can be placed in front or behind another window, its size can be adjusted, and scrollbars can be used to navigate the sections within it. Multiple windows can also be opened at one time, in which case each window can display a different application or file – this is very useful when working in a multitasking environment. The system's memory is the only limitation to the number of windows that can be opened at once. There are also many types of specialised windows.

- A **Container Window**: a window that is opened while invoking the icon of a mass storage device, directory or folder and which is presenting an ordered list of other icons that could be again some other directories, or data files or may be even executable programs. All modern container windows could present their content on screen either by acting as browser windows or text windows. Their behaviour can automatically change according to the choices of the single users and their preferred approach to the graphical user interface.
- A **browser window**: allows the user to move forward and backward through a sequence of documents or web pages. Web browsers are examples of these types of windows.
- **Text terminal windows**: are designed for embedding interaction with text user interfaces within the overall graphical interface.

MS-DOS and UNIX consoles are examples of these types of windows.

- A **child window**: opens automatically or as a result of a user activity in a parent window (A parent window can be any type of window). Pop-up windows on the Internet can be child windows.
- A **message window**, or **dialog box**: is a type of child window. These are usually small and basic windows that are opened by a program to display information to the user and/or get information from the user. They usually have a button that must be pushed before the program can be resumed.

## Menus

Menus allow the user to execute commands by selecting from a list of choices. Options are selected with a mouse or other pointing device within a GUI. A keyboard may also be used. Menus are convenient because they show what commands are available within the software. This limits the amount of documentation the user reads to understand the software.

- A **menu bar** is displayed horizontally across the top of the screen and/or along the top of some or all windows. A pull-down menu is commonly associated with this menu type. When a user clicks on a menu option, the pull-down menu will appear.
- A **menu** has a visible title within the menu bar. Its contents are only revealed when the user selects it with a pointer. The user is then able to select the items within the pull-down menu. When the user clicks elsewhere, the contents of the menu will disappear.
- A **context menu** is invisible until the user performs a specific mouse action, like pressing the right mouse button. When the software-specific mouse action occurs the menu will appear under the cursor.
- **Menu extras** are individual items within or at the side of a menu.

## Icons

An icon is a small picture that represents objects such as a file, program, web page, or command. Icons are used as a quick way to execute commands, open documents, and run programs. They are also very useful when searching for an object in a browser list, because in many operating systems all documents using the same extension will have the same icon.

## Controls (or Widgets)

The interface element that a computer user interacts with is known as a **control** or **widget**.

### *Window*

A paper-like rectangle that represents a "window" into a document, form, or design area.

### *Pointer (or mouse cursor)*

The spot where the mouse "cursor" is currently referencing.

### *Text box*

A box in which texts or numbers are entered.

### *Button*

An equivalent to a push-button as found on mechanical or electronic instruments.

### *Hyperlink*

Text with some kind of indicators (usually underlining and/or colour) that shows that clicking it will take one to another screen or page.

### *Drop-down list*

A list of items from which to select: The list normally only displays items when a special button or indicator is clicked.

### *Check box*

A box which indicates an "on" or "off" state via a check-mark or an "×".

### *Radio button*

A button, similar to a check-box, except that only one item in a group can be selected. Its name comes from the mechanical push-button group on a car radio receiver. Selecting a new item from the group's buttons also deselects the previously selected button.

### *Data grid*

A spreadsheet-like grid that allows numbers or text to be entered in rows and columns.

## **Tabs**

A tab is typically a rectangular small box which usually contains a text label or graphical icon associated with a view pane. When activated the view pane, or window, displays widgets associated with that tab; groups of tabs allow the user to switch quickly between different widgets. This

is used in the web browsers Firefox, Internet Explorer, Konqueror, Opera, and Safari. With these browsers, you can have multiple web pages open at once in one window, and quickly navigate between them by clicking on the tabs associated with the pages. Tabs are usually placed in groups at the top of a window, but may also be grouped on the side or bottom of a window.

### **3.3.2 Interaction Elements**

Some common idioms for interaction have evolved in the visual language used in GUIs. Interaction elements are interface objects that represent the state of an ongoing operation or transformation, either as visual reminders of the user intent (such as the pointer), or as affordances showing places where the user may interact.

#### **Cursor**

A cursor is an indicator used to show the position on a computer monitor or other display devices that will respond to inputs from a text input or pointing devices.

#### **Pointer**

One of the most common components of a GUI on the personal computer is a pointer: a graphical image on a screen that indicates the location of a pointing device, and can be used to select and move objects or commands on the screen. A pointer commonly appears as an angled arrow, but it can vary within different programs or operating systems. Example of this can be found within text-processing applications, which use an I-beam pointer that is shaped like a capital I, or in web browsers, which often indicate that the pointer is over a hyperlink by turning the pointer in the shape of a gloved hand with outstretched index finger.

The use of a pointer is employed when the input method, or pointing device, is a device that can move fluidly across a screen and select or highlight objects on the screen. Pointer trails can be used to enhance its visibility during movement. In GUIs where the input method relies on hard keys, such as the five-way key on many mobile phones, there is no pointer employed, and instead the GUI relies on a clear focus state.

#### **Selection**

A selection is a list of items on which user operations will take place. The user typically adds items to the list manually, although the computer may create a selection automatically.

## Adjustment handle

A handle is an indicator of a starting point for a drag and drop operation. Usually the pointer shape changes when placed on the handle, showing an icon that represents the supported drag operation.

## SELF-ASSESSMENT EXERCISE 1

Identify and study these elements within Window operating system

### 3.4 Three-dimensional User Interfaces

For typical computer displays, *three-dimensional* are a misnomer—their displays are two-dimensional. Three-dimensional images are projected on them in two dimensions. Since this technique has been in use for many years, the recent use of the term three-dimensional must be considered a declaration by equipment marketers that the speed of three dimensions to two dimensions projection is adequate to use in standard GUIs.

#### Motivation

Three-dimensional GUIs are quite common in science fiction literature and movies, such as in *Jurassic Park*, which features Silicon Graphics' three-dimensional file manager, "File system navigator", an actual file manager that never got much widespread use as the user interface for a Unix computer. In fiction, three-dimensional user interfaces are often immersible environments like William Gibson's *Cyberspace* or Neal Stephenson's *Metaverse*.

Three-dimensional graphics are currently mostly used in computer games, art and computer-aided design (CAD). There have been several attempts at making three-dimensional desktop environments like Sun's Project Looking Glass or SphereXP from Sphere Inc. A three-dimensional computing environment could possibly be used for collaborative work. For example, scientists could study three-dimensional models of molecules in a virtual reality environment, or engineers could work on assembling a three-dimensional model of an airplane. This is a goal of the Croquet project and Project Looking Glass.

#### Technologies

The use of three-dimensional graphics has become increasingly common in mainstream operating systems, from creating attractive interfaces—eye candy—to functional purposes only possible using three dimensions. For example, user switching is represented by rotating

a cube whose faces are each user's workspace, and window management is represented in the form or via a Rolodex-style flipping mechanism in Windows Vista (see Windows Flip 3D). In both cases, the operating system transforms windows on-the-fly while continuing to update the content of those windows.

Interfaces for the X Window System have also implemented advanced three-dimensional user interfaces through compositing window managers such as Beryl, Compiz and KWin using the AIGLX or XGL architectures, allowing for the usage of OpenGL to animate the user's interactions with the desktop.

Another branch in the three-dimensional desktop environment is the three-dimensional GUIs that take the desktop metaphor a step further, like the BumpTop, where a user can manipulate documents and windows as if they were "real world" documents, with realistic movement and physics.

The Zooming User Interface (ZUI) is a related technology that promises to deliver the representation benefits of 3D environments without their usability drawbacks of orientation problems and hidden objects. It is a logical advancement on the GUI, blending some three-dimensional movement with two-dimensional or "2.5D" vector objects.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to graphical user interface (GUI). The history of graphical user interface was also discussed. The elements of graphical user interface were also explained. You were also introduced to three-dimensional user interfaces.

#### **5.0 SUMMARY**

In this unit, you have learnt the:

- introduction to graphical user interface which is a type of user interface that allows people to interact with electronic devices such as computers, hand-held devices, household appliances and office equipment with images rather than text commands
- history of graphical user interface, precursor to GUI, PARC user interface and the evolution of other graphical user interfaces
- elements of graphical user interface which are divided into two categories that includes structural and interactive elements.

## **SELF-ASSESSMENT EXERCISE 2**

- i. What do you understand by graphical user interface?
- ii. Explain the structural and iterative elements of graphical user interface.

## **6.0 TUTOR- MARKED ASSIGNMENT**

Explain the PARC graphical user interface

## **7.0 REFERENCES/FURTHER READING**

<http://www.linfo.org/gui.html>. Retrieved on 12 November 2008.

[www.wikipedia.com](http://www.wikipedia.com)



## UNIT 4 HUMAN- COMPUTER INTERACTION

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Human -Computer Interaction (HCI)
  - 3.2 Goals of HCI
  - 3.3 Differences with Other Related Fields
  - 3.4 Future Development of HCI
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor -Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In this unit, you will be introduced to Human-Computer Interaction (HCI) and its differences with other related fields. The goals and future development of human- computer interaction and the general concept of human-computer interface will be introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the term human- computer interaction
- identify the various goals of human -computer interaction
- differentiate human -computer interaction from other related fields
- describe the future development of HCI and explain the human - computer interface.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Human -Computer Interaction

**Human–Computer Interaction (HCI)** is the study of interaction between people (users) and computers. It involves the study intersection of computer science, behavioural sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply *interface*), which includes both software and hardware. The association for computing machinery defines *human-computer interaction* as "a discipline concerned with the design,

evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them” Since human-computer interaction studies a human and a machine together, it draws its supporting knowledge from the machine and human sides. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, statistics, linguistics, social sciences, cognitive psychology, and human performance are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success. HCI is sometimes referred to as **Man–Machine Interaction (MMI)** or **Computer–Human Interaction (CHI)**.

### 3.1.1 Human–Computer Interface

The human–computer interface can be described as the point of communication between the human user and the computer. The flow of information between the human and computer is defined as the loop of interaction. The loop of interaction has several aspects to it including:

- **Task Environment:** The conditions and goals set upon the user.
- **Machine Environment:** The environment that the computer is connected to, i.e. a laptop in a college student's dorm room.
- **Areas of the Interface:** Non-overlapping areas involve processes of the human and computer not pertaining to their interaction. Meanwhile, the overlapping areas only concern themselves with the processes pertaining to their interaction.
- **Input Flow:** Begins in the task environment as the user has some task that requires using their computer.
- **Output:** The flow of information that originates in the machine environment.
- **Feedback:** Loops through the interface that evaluate, moderate, and confirm processes as they pass from the human through the interface to the computer and back.

### 3.2 Goals of HCI

The basic goal of HCI is to improve the interactions between users and computers by making computers more usable and responsive to the user's needs. Specifically, HCI is concerned with:

- methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface

within given constraints, optimising for a desired property such as learning ability or efficiency of use)

- methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- techniques for evaluating and comparing interfaces
- developing new interfaces and interaction techniques
- developing descriptive and predictive models and theories of interaction.

A long term goal of HCI is to design systems that minimise the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

Researchers in HCI are interested in developing new design methodologies, experimenting with new hardware devices, prototyping new software systems, exploring new paradigms for interaction, and developing models and theories of interaction.

### **3.3 Differences with Related Fields**

HCI focuses on user interface design mainly for users of computer system and promotes effective interaction between computers and users (human). User Interface Design is concerned with the users of devices such as computers, appliances, machines, mobile communication devices, software applications, and websites. In HCI, efficient user interface is critical.

HCI differs from human factors in that the focus is more on users working specifically with computers, rather than other kinds of machines or designed artifacts. There is also a focus in HCI on how to implement the computer software and hardware mechanisms to support human-computer interaction. Thus, human factors are a broader term; HCI could be described as the human factors of computers, although some experts try to differentiate these areas.

According to some experts, HCI also differs from ergonomics (which will be fully introduced in the next unit) in that there is less focus on repetitive work-oriented tasks and procedures, and much less emphasis on physical stress and the physical form or industrial design of the user interface, such as keyboards and mice. However, this does not take full account of ergonomics, which recently has gained a much broader focus (equivalent to human factors). Cognitive ergonomics, for example, is a

part of ergonomics, of which *software ergonomics* (an older term, essentially the same as HCI) is a part.

Three areas of study have substantial overlap with HCI even as the focus of inquiry shifts. In the study of Personal Information Management (PIM), human interactions with the computer are placed in a larger informational context - people may work with many forms of information, some computer-based, many not (e.g., whiteboards, notebooks, sticky notes, refrigerator magnets) in order to understand and effect desired changes in their world. In Computer Supported Cooperative Work (CSCW), emphasis is placed on the use of computing systems in support of the collaborative work of a group of people. The principles of Human Interaction Management (HIM) extend the scope of CSCW to an organisational level which can be implemented without the use of computer systems.

### 3.4 Future Development of HCI

As the means by which humans interact with computers continues to evolve rapidly, human-computer interaction is affected by the forces shaping the nature of future computing. These forces include:

- decreasing hardware costs leading to larger memories and faster systems
- miniaturisation of hardware leading to portability
- reduction in power requirements leading to portability
- new display technologies leading to the packaging of computational devices in new forms
- specialised hardware leading to new functions
- increased development of network communication and distributed computing
- increasingly widespread use of computers, especially by people who are outside the computing profession
- increasing innovation in input techniques (i.e., voice, gesture, pen), combined with lowering cost, leading to rapid computerisation by people previously left out of the "computer revolution"
- wider social concerns leading to improved access to computers by currently disadvantaged groups.

The future for HCI is expected to include the following features:

**Ubiquitous communication:** Computers will communicate through high speed local networks, nationally over wide-area networks, and portably via infrared, ultrasonic, cellular, and other technologies. Data

and computational services will be portably accessible from many if not most locations to which a user travels.

**High functionality systems:** Systems will have large numbers of functions associated with them. There will be so many systems that most users, technical or non-technical, will not have time to learn them in the traditional way (e.g., through thick manuals).

**Mass availability of computer graphics:** Computer graphics capabilities such as image processing, graphics transformations, rendering, and interactive animation will become widespread as inexpensive chips become available for inclusion in general workstations.

**Mixed media:** Systems that will handle images, voice, sounds, video, text, formatted data. These will be exchangeable over communication links among users. The separate worlds of consumer electronics (e.g., stereo sets, VCRs, televisions) and computers will partially merge. Computer and print worlds will continue to cross assimilate each other.

**High-bandwidth interaction:** The rate at which humans and machines interact will increase substantially due to the changes in speed, computer graphics, new media, and new input/output devices. This will lead to some qualitatively different interfaces, such as virtual reality or computational video.

**Large and thin displays:** New display technologies will finally mature enabling very large displays and also displays that are thin, light weight, and have low power consumption. This will have large effects on portability and will enable the development of paper-like, pen-based computer interaction systems very different in feel from desktop workstations of the present.

**Embedded computation:** Computation will pass beyond desktop computers into every object for which uses can be found. The environment will be alive with little computations from computerised cooking appliances. to lighting and plumbing fixtures to window blinds to automobile braking systems to greeting cards. To some extent, this development is already taking place. The difference in the future is the addition of networked communications that will allow many of these embedded computations to coordinate with each other and with the user. Human interfaces to these embedded devices will in many cases be very different from those appropriate to workstations.

**Augmented reality:** A common staple of science fiction, augmented reality refers to the notion of layering relevant information into our

vision of the world. Existing projects show real-time statistics to users performing difficult tasks, such as manufacturing. Future work might include augmenting our social interactions by providing additional information about those we converse with.

**Group interfaces:** Interfaces to allow groups of people to coordinate will be common (e.g., for meetings, for engineering projects, for authoring joint documents). These will have major impacts on the nature of organisations and on the division of labour. Models of the group design process will be embedded in systems and will cause increased rationalisation of the design.

**User tailorability:** Ordinary users will routinely tailor applications to their own use and will use this power to invent new applications based on their understanding of their own domains. Users, with their deeper knowledge of their own knowledge domains, will increasingly be important sources of new applications at the expense of generic systems programmers (with systems expertise but low domain expertise).

**Information utilities:** Public information utilities (such as home banking and shopping) and specialised industry services (e.g., weather for pilots) will continue to proliferate. The rate of proliferation will accelerate with the introduction of high-bandwidth interaction and the improvement in quality of interfaces.

## 4.0 CONCLUSION

In this unit, you have been introduced to the concepts of human - computer interaction. You have also been introduced to the various goals of HCI and also the difference between HCI and other related fields. The future of HCI was also discussed. The concept of the human-computer interface was also introduced.

## 5.0 SUMMARY

In this unit, you have learnt the following:

- introduction to human-computer interaction (HCI) which is the study of interaction between people (users) and computers. HCI is also sometimes referred to as man-machine interaction (MMI) or computer-human interaction (CHI)
- the human-computer interface which can be described as the point of communication between the human user and the computer

- the goal of HCI which is basically to improve the interactions between users and computers by making computers more usable and receptive to the user's needs
- the difference between HCI and other related fields like graphical user interface, ergonomics e.t.c.
- the future development in HCI like Ubiquitous communication, High functionality systems, Mass availability of computer graphics e.t.c.

### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by Human -Computer Interaction?
- ii. Describe the Human -Computer Interface

### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly the future development in HCI.

### **7.0 REFERENCES/FURTHER READING**

More discussion of the differences between these terms can be found in the ACM SIGCHI Curricula for Human-Computer Interaction

Brown, C. M. (1998). *Human-Computer Interface Design Guidelines*. Intellect Books. pp.2–3.

Green, P. (2008). Iterative Design. Lecture presented in Industrial and Operations Engineering 436 Human Factors in Computer Systems, University of Michigan, Ann Arbor, MI, February 4, 2008.

Wickens, C. D., John, D. L, Yili, L., & Sallie, E. G. (2004). *An Introduction to Human Factors Engineering*. (2nd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. pp.185–193.

## **UNIT 5     ERGONOMICS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Ergonomics
  - 3.2 Five Aspects of Ergonomics
  - 3.3 History of Ergonomics
  - 3.4 Ergonomics in Workplace
  - 3.5 Efficiency and Ergonomics
  - 3.6 Benefits of Ergonomics
  - 3.7 Fields of Ergonomics
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

This unit describes in great detail the various aspects of ergonomics alongside the history behind it. The efficiency and benefits of ergonomics will also be discussed. Different fields of ergonomics will also be highlighted.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- define the term ergonomics
- identify the various aspects of ergonomics
- explain the history of ergonomics
- describe efficiency and ergonomics
- identify the various benefits of ergonomics
- list the various fields of ergonomics.

### **3.0 MAIN CONTENT**

#### **3.1 Introduction to Ergonomics**

Ergonomics is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work.



The International Ergonomics Association has adopted this technical definition: ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimise human well-being and overall system performance.

Ergonomics is the science of making things comfy. It also makes things efficient. However for simplicity, ergonomics makes things comfortable and efficient.

At its simplest definition, ergonomics literally means the science of work. So ergonomists, i.e. the practitioners of ergonomics, study work, how work is done and how to work better. It is the attempt to make work better that ergonomics becomes so useful.

However, what you, or the user, is most concerned with is, “How can I use the product or service, will it meet my needs, and will I like using it?” Ergonomics helps define how it is used, how it meets you needs, and most importantly if you like it. It makes things comfy and efficient.

Ergonomics is concerned with the ‘fit’ between people and their work. It takes account of the worker's capabilities and limitations in seeking to ensure that tasks, equipment, information and the environment suit each worker.

To assess the fit between a person and his/her work, ergonomists consider the job being done and the demands on the worker; the equipment used (its size, shape, and how appropriate it is for the task), and the information used (how it is presented, accessed, and changed). Ergonomics draws on many disciplines in its study of humans and their environments, including anthropometry, biomechanics, mechanical engineering, industrial engineering, industrial design, kinesiology, physiology and psychology.

### **3.2 Five Aspects of Ergonomics**

There are five aspects of ergonomics: safety, comfort, ease of use, productivity/performance, and aesthetics. From these aspects of ergonomics, examples are given of how products or systems could benefit from redesign based on ergonomic principles.

- Safety – This has to do with the ability to use a device or work with a device without short or long term damage to parts of the body. For example in Medicine bottles: The print on them could be larger so that a sick person who may have bad vision (due to

sinuses, etc.) can easily see the dosages and label. Ergonomics could design the print style, colour and size for optimal viewing.

- Comfort – Comfort in the human-machine interface is usually noticed first. Physical comfort in how an item feels is pleasing to the user. If you do not like to touch it you won't. If you do not touch it you will not operate it. If you do not operate it, then it is useless. For example, in Alarm clock display: Some displays are harshly bright, drawing one's eye to the light when surroundings are dark. Ergonomic principles could re-design this based on contrast principles.
- Ease of use – This has to do with the ability to use a device with no stress. For example in Street Signs: In a strange area, many times it is difficult to spot street signs. This could be addressed with the principles of visual detection in ergonomics.
- Productivity/performance – For example in High-definition Television (HD TV): The sound on HD TV is much lower than regular TV. So when you switch from HD to regular, the volume increases dramatically. Ergonomics recognises that this difference in decibel level creates a difference in loudness and hurts human ears and this could be solved by evening out the decibel levels.
- Aesthetics - the look and feel of the object, the user experience.

### 3.3 History of Ergonomics

The foundations of the science of ergonomics appear to have been laid within the context of the culture of Ancient Greece. A good deal of evidence indicates that Hellenic civilisation in the 5th century BC used ergonomic principles in the design of their tools, jobs, and workplaces.

The term ergonomics is derived from the Greek words *ergon* [work] and *nomos* [natural laws] and first entered the modern lexicon when Wojciech Jastrzębowski used the word in his 1857 article *Rys ergonomji czyli nauki o pracy, opartej na prawdach poczerpniętych z Nauki Przyrody* (The Outline of Ergonomics, i.e. Science of Work, Based on the Truths Taken from the Natural Science).

Later, in the 19th century, Frederick Winslow Taylor pioneered the "Scientific Management" method, which proposed a way to find the optimum method for carrying out a given task. Taylor found that he could, for example, triple the amount of coal that workers were shoveling by incrementally reducing the size and weight of coal shovels

until the fastest shoveling rate was reached. Frank and Lillian Gilbreth expanded Taylor's methods in the early 1900s to develop "Time and Motion Studies". They aimed at improving efficiency by eliminating unnecessary steps and actions. By applying this approach, the Gilbreths reduced the number of motions in bricklaying from 18 to 4.5, allowing bricklayers to increase their productivity from 120 to 350 bricks per hour.

World War II marked the development of new and complex machines and weaponry, and these made new demands on operators' cognition. The decision-making, attention, situational awareness and hand-eye coordination of the machine's operator became keys to the success or failure of a task. It was observed that fully functional aircraft, flown by the best-trained pilots, still crashed. In 1943, Alphonse Chapanis, a lieutenant in the U.S. Army, showed that this so-called "pilot error" could be greatly reduced when more logical and differentiable controls replaced confusing designs in airplane cockpits. The experience of the World War II promoted the study. Man-Machine interaction became a major focus when weapons meant to attack the enemies were malfunctioning – attacking friends instead of foes.

In the decades since the war, ergonomics has continued to flourish and diversify. The Space Age created new human factors issues such as weightlessness and extreme g-forces. How far could environments in space be tolerated, and what effects would they have on the mind and body? The dawn of the Information Age has resulted in the new ergonomics field of human-computer interaction (HCI). Likewise, the growing demand for and competition among consumer goods and electronics has resulted in more companies including human factors in product design.

At home, work, or play new problems and questions must be resolved constantly. People come in all different shapes and sizes, and with different capabilities and limitations in strength, speed, judgment, and skills. All of these factors need to be considered in the design function. To solve design problems, physiology and psychology must be included with an engineering approach.

### 3.4 Ergonomics in Workplace



**Fig. 5.1: Description of Workplace Environment**

(Source: [www.wikipedia.org](http://www.wikipedia.org))

Fundamentals for the Flexible Workplace variability and compatibility with desk components, that flex from individual work activities to team settings. Workstations provide supportive ergonomics for task-intensive environments.

Outside of the discipline itself, the term 'ergonomics' is generally used to refer to physical ergonomics as it relates to the workplace (example ergonomic chairs and keyboards). Ergonomics in the workplace has to do largely with the safety of employees, both in the long and short-term. Ergonomics can help reduce costs by improving safety. This would decrease the money paid out in workers' compensation.

Workplaces may either take the reactive or proactive approach when applying ergonomics practices. Reactive ergonomics is when something needs to be fixed, and corrective action is taken. Proactive ergonomics is the process of seeking areas that could be improved and fixing the issues before they become a large problem. Problems may be fixed through equipment design, task design, or environmental design. Equipment design changes the actual, physical devices used by people. Task design changes what people do with the equipment. Environmental design changes the environment in which people work, but not the physical equipment they use.

### 3.5 Efficiency and Ergonomics

Efficiency is quite simply making something easier to do. Several forms of efficiency are:

- Reducing the strength required makes a process more physically efficient.
- Reducing the number of steps in a task makes it quicker (i.e. efficient) to complete.
- Reducing the number of parts makes repairs more efficient.
- Reducing the amount of training needed, i.e. making it more intuitive, gives you a larger number of people who are qualified to perform the task. Imagine how in-efficient trash disposal would be if your teenage child wasn't capable of taking out the garbage. Have you tried an ergonomic trash bag?

Efficiency can be found almost everywhere. If something is easier to do you are more likely to do it. If you do it more, then it is more useful. Again, utility is the only true measure of the quality of a design. And if you willingly do something more often you have a greater chance of liking it. If you like doing it you will be more comfortable doing it. So the next time you hear the term ergonomics you will know what it means to you. And I hope that is a comforting thought.

Ergonomics can help you in many ways. Among other things, it can benefit your life, health, productivity and accuracy. One of the best benefits of ergonomics is saving time. We never seem to have enough of it as it is, so why not try to get a little more out of your day? Ergonomics is about making things more efficient. By increasing the efficiency of a tool or a task, you tend to shorten the length of time it takes to accomplish your goal.

### **3.6 Benefits of Ergonomics**

The three main benefits of ergonomics are:

#### **Slim Down the Task**

Have you ever wondered why some things are so convoluted, cumbersome and chaotic? And they take forever to complete. And most of what you do does not aid the outcome.

For example, think back to the last time you got hired for a job, bought a house or car, or did something else that required a ton of paperwork. How many different forms did you write the same information on? That was not very ergonomic.

You can almost always make a task a little leaner. But first you have to understand the task. Ergonomics requires that tasks are properly slimed down and steps involved in a task are well written out. This is why task

analysis is normally done in ergonomics (Task analysis is fully discussed in module 2, unit 2).

Once you have all the steps written out, you need to take a good look at them and identify areas that you can "ergonomise":

1. *Repetition* – Look for steps that are repeated and see if they are all necessary.
2. *Order* – See if you can re-order the steps to optimise your effort.
3. *Synergy* – Can you combine things or somehow get more bang for your buck?
4. *Value Added* – Look at every step and make sure it adds value to the outcome. If it doesn't, cut it.
5. *Necessity* – Make sure the quantity of the step is needed. Do you really need to brush your teeth with 57 strokes, or will 32 do?

### **Simplify the Task**

You can also save time by simplifying the task. This is not just about reducing the number of steps, but making those steps easier to perform. The less training and/or skill that is required for a task, the quicker the pace at which it tends to get finished.

This is a great ergonomic tip, especially when the task requires more than one person. If you are trying to get your kids to pick up their toys before they go to bed, you can save a lot of time by making it easier. That is what the toy chest is for. Instead of having different places for different things, they can just throw everything in one place.

### **Increase Body Mechanism**

Ergonomic can increase your body mechanics. A good ergonomic tool acts as extension of your body enhancing capabilities. Some tools make you more effective and faster at completing a task. (Try cutting a log without an axe and see how long it takes you.)

## **3.7 Fields of Ergonomics**

### **Engineering Psychology**

*Engineering psychology* is an interdisciplinary part of Ergonomics and studies the relationships of people to machines, with the intent of improving such relationships. This may involve redesigning equipment, changing the way people use machines, or changing the location in which the work takes place. Often, the work of an engineering

psychologist is described as making the relationship more "user-friendly."

Engineering Psychology is an applied field of psychology concerned with psychological factors in the design and use of equipment. Human factors are broader than engineering psychology, which is focused specifically on designing systems that accommodate the information-processing capabilities of the brain.

### **Macroergonomics**

Macroergonomics is an approach to ergonomics that emphasises a broad system view of design, examining organisational environments, culture, history, and work goals. It deals with the physical design of tools and the environment. It is the study of the society/technology interface and their consequences for relationships, processes, and institutions. It also deals with the optimisation of the designs of organisational and work systems through the consideration of personnel, technological, and environmental variables and their interactions. The goal of macroergonomics is a completely efficient work system at both the macro- and micro-ergonomic level which results in improved productivity, and employee satisfaction, health, safety, and commitment. It analyses the whole system, finds how each element should be placed in the system, and considers all aspects for a fully efficient system. A misplaced element in the system can lead to total failure.

### **Seating Ergonomics**

The best way to reduce pressure in the back is to be in a standing position. However, there are times when you need to sit. When sitting, the main part of the body weight is transferred to the seat. Some weight is also transferred to the floor, backrest, and armrests. Where the weight is transferred is the key to a good seat design. When the proper areas are not supported, sitting in a seat all day can put unwanted pressure on the back causing pain.

The lumbar (bottom five vertebrae in the spine) needs to be supported to decrease disc pressure. Providing both a seat back that inclines backwards and has a lumbar support is critical to prevent excessive low back pressures. The combination which minimises pressure on the lower back is having a backrest inclination of 120 degrees and a lumbar support of 5 cm. The 120 degrees inclination means the angle between the seat and the backrest should be 120 degrees. The lumbar support of 5 cm means the chair backrest supports the lumbar by sticking out 5 cm in the lower back area.

Another key to reducing lumbar disc pressure is the use of armrests. They help by putting the force of your body not entirely on the seat and backrest, but putting some of this pressure on the armrests. Armrest needs to be adjustable in height to assure that shoulders are not overstressed.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the fundamental concepts of Ergonomics. You have also learnt the different aspect of ergonomics and also the history of ergonomics. Ergonomics in workplace was also discussed alongside achieving efficiency in ergonomics. The various benefits of ergonomics were also discussed. The various fields of ergonomics were also briefly explained.

#### **5.0 SUMMARY**

In this unit you, have learnt the following:

- introduction to ergonomics which is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work
- highlighting of the various aspect of ergonomics like safety, comfort, ease of use e.t.c.
- the history of ergonomics whose foundations appears to have been laid within the context of the culture of Ancient Greece
- the discussion of ergonomics in workplace and achieving efficiency in ergonomics
- explanation of the various benefits of ergonomics which was discussed in greater detail
- the discussion of various fields of ergonomics like engineering psychology, Macroergonomics, Seating ergonomics.

#### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by ergonomics?
- ii. Highlight the various benefits of ergonomics, giving examples.

#### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly any two fields of ergonomics.



## 7.0 REFERENCES/FURTHER READING

Berkeley Lab. *Integrated Safety Management: Ergonomics*. Website. Retrieved 9 July 2008.

Berkeley Lab. *Today at Berkeley Lab: Ergonomic Tips for Computer Users*. Retrieved 8 January 2009.

Brookhuis, K., Hedge, A., Hendrick, H., Salas, E., & Stanton, N. (2005). *Handbook of Human Factors and Ergonomics Models*. Florida: CRC Press.

Wickens & Hollands (2006). *Engineering Psychology and Human Performance*.

[www.wikipedia.com](http://www.wikipedia.com)

## **MODULE 1 INTRODUCTION TO USER INTERFACE DESIGN**

Unit 1	Fundamentals of User Interface Design
Unit 2	Designing Good User Interfaces
Unit 3	Graphical User Interface (GUI)
Unit 4	Human-Computer Interaction
Unit 5	Ergonomics

### **UNIT 1 FUNDAMENTALS OF USER INTERFACE DESIGN**

#### **CONTENTS**

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	User Interface
3.2	Significance of User Interface
3.3	Types of User Interfaces
3.4	History of User Interfaces
3.5	User Interface Modes and Modalities
3.6	Introduction to Usability
4.0	Conclusion
5.0	Summary
6.0	Tutor- Marked Assignment
7.0	References/Further Reading

#### **1.0 INTRODUCTION**

Having read through the course guide, you will have a general understanding of what this unit is about and how it fits into the course as a whole. This unit describes the general fundamentals of user interface design.

#### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain the term user interface design
- identify the significance of user interface
- explain the history behind user interfaces
- describe the modalities and modes of user interface.

## 3.0 MAIN CONTENT

### 3.1 User Interface

The **user interface** (also known as Human Machine Interface (HMI) or Man-Machine Interface (MMI)) is the aggregate of means by which people—the *users*—interact with the *system*—a particular machine, device, computer program or other complex tool.

**User interface** is the point at which a user or a user department or organisation interacts with a computer system. The part of an interactive computer program that sends messages to and receives instructions from a terminal user.

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

#### **User Interface Design or User Interface Engineering**

This is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centred design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design may be utilised to apply a theme or style to the interface without compromising its usability. The design process must balance the technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialise in certain types of projects and have skills centered around their expertise, e.g., software design, user research, web design, or industrial design.

### 3.2 Significance of User Interface

To work with a system, users have to be able to control the system and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windscreen and the exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

The term *user interface* is often used in the context of computer systems and electronic devices. The user interface of a mechanical system, a vehicle or an industrial installation is sometimes referred to as the **Human-Machine Interface (HMI)**. HMI is a modification of the original term MMI (Man-Machine Interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, which is more commonly used for Human-Computer *Interaction* than Human-Computer *Interface*. Other terms used are Operator Interface Console (OIC) and Operator Interface Terminal (OIT).

### 3.3 Types of User Interfaces

Currently (as at 2009) the following types of user interface are the most common:

- **Graphical User Interfaces (GUI)** accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-Oriented User Interfaces (OOUIs) and application oriented interfaces. Examples are Windows Operating System, LabView, etc.
- **Web-based User Interfaces or Web User Interfaces (WUI)** accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilise Java, AJAX, Adobe Flex, Microsoft, NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called control panels.

User interfaces that are common in various fields outside desktop computing:

- **Command Line Interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Examples are MS-DOS and UNIX interface. This interface is also used for system administration tasks, etc.
- **Tactile Interfaces** supplement or replace other forms of output with haptic feedback methods. This is used in automated simulators and so on.
- **Touch User Interface** are graphical user interfaces using a touchscreen display as a combined input and output device. It is used in many types of point of sale, industrial processes and machines, self-service machines etc. Examples are touch screen monitors.

Other types of user interfaces are:

- **Attentive User Interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch Interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based Interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture Interface** are graphical user interfaces which accept input in form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent User Interfaces** are human-machine interfaces that aim at improving the efficiency, effectiveness, and naturalness of

human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).

- **Motion Tracking Interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple.
- **Multi-screen Interfaces** employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Non-command User Interfaces** allow users to infer their needs and intentions, without requiring explicit commands formulation.
- **Object-Oriented User Interface (OOUI)** examples of OOUI are:

**Reflexive User Interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.

**Tangible User Interfaces** which place a greater emphasis on touch and physical environment or its element.

**Text User Interfaces** are user interfaces which output text, but accept other forms of input in addition to or in place of typed command strings.

**Voice User Interfaces** which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.

**Natural-Language Interfaces** used for search engines and on webpages. User types in a question and waits for a response.

**Zero-Input Interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.

**Zooming User Interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail

## SELF-ASSESSMENT EXERCISE

Check the GUI features and functions of LabView. Then compare their environments with MS-DOS.

### 3.4 History of User Interfaces

The history of user interfaces can be divided into the following phases according to the dominant type of user interface:

- Batch Interface, 1945-1968
- Command-line User Interface, 1969 to present
- Graphical User Interface, 1981 to present

### 3.5 User Interface Modalities and Modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

Input — allowing the users to manipulate a system. For example the computer keyboard allows the user to enter typed text; digitising tablet allows the user to create free-form drawing.

Output — allowing the system to indicate the effects of the users' manipulation. For example the computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*).

The user interface may employ several redundant input and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending on the state of the computer program. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

## 4.0 CONCLUSION

In this unit, you have been introduced to the fundamental concepts of user interface. You have also learnt the history and significance of user interface design.

## 5.0 SUMMARY

In this unit, you have learnt the:

- introduction of user interface which is the aggregate of means by which users interact with a particular machine, device, computer program or any other complex tool
- study of the various types of user interface design which includes graphical user interfaces, web-based user interfaces, command line interfaces e.t.c.
- history of user interfaces which can be divided into batch interface, command-line user interface and graphical user interface
- modality of a user interface which is a path of communication employed by the user interface to carry input and output.

## 6.0 TUTOR-MARKED ASSIGNMENT

- i. Explain the Microsoft's Clippy the paperclip.
- ii. Write a short note on the Command-line User Interface.

## 7.0 REFERENCES/FURTHER READING

Pinel, J. P. (2008). *Biopsychology* (7th ed.). Boston: Pearson. p. 357.  
Wikipedia.org



## UNIT 2 DESIGNING GOOD USER INTERFACES

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Essentials of Interface Design
    - 3.1.1 Designing a Good Interface
    - 3.1.2 Tips for Designing Good User Interface
  - 3.2 Understanding Users
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the essentials of designing good interface designs and also discusses the various users.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the essentials of a good interface design
- identify the necessary tips needed for designing a good interface
- discuss various users.

### 3.0 MAIN CONTENT

#### 3.1 Essentials of Interface Design

There are three pillars to an application's success:

- Features
- Function
- Face

*Features* refer to what the application will do for the user. Features are the requirements for the software.

*Function* refers to how well the software operates. Bug-free software will function perfectly.

**Face** refers to how the application presents itself to the user; the application's "user interface."

Features, function and face can be restated as questions:

- i. Does the software meet the user's requirements? (Features)
- ii. Does the software operate as intended? (Function)
- iii. Is the software easy to use? (Face)

The goal of user interface design is to put a happy face on the application. Stated in more concrete terms, a successful user interface will require *Zero Training* and will be *friendly not foreboding*.

### ***Zero Training***

The goal of Zero Training could be considered as a sub-goal of friendly not foreboding. However, training costs are a major impediment to the usage of software making Zero Training an important goal by itself.

There are two types of training involved in software design: software training and job training. Software training assumes the user knows how to do the job at hand and only needs to learn how to use the software to do the job. Job training teaches the user how to do the job - which can include more than how to use the application to do the job. The goal of Zero Training relates to *zero software training*. Job training can be integrated with software training, but results in a much more ambitious project.

### ***Friendly not Foreboding***

Almost everything you do to implement the goal of Zero Training will further the goal of being friendly not foreboding. However, some techniques for reducing training may slow up experienced users. For example, you could pop-up new user messages whenever the user lands in a particular field. Seeing the same message after awhile makes the experienced user dispense with the messages.

Being friendly is an attitude and encompasses more than what is necessary for the Zero Training goal. Applications do have an attitude. For example, consider the following sets of application messages:

"Database does not exist" "I could not find database "CorpInfo".  
If you are sure this name is correct,  
CorpInfo could be unavailable due to  
maintenance or LAN problems. You

should contact the help desk to see when CorpInfo will again be available.”

“SQL access error 123” “I could not save information to the database. You can try to save again to see if the error clears. If you call the help desk concerning this problem, tell them you have a “SQL access error 123”.

“out of hunk”<sup>1</sup> “I have run out of memory (RAM). This typically happens when there is a bug in the program which causes it to lose memory over time. Save your game if possible. To free the memory you will need to reset the computer (turn it off and then on).”

The attitude of the first message is “you did something that caused me a problem” while the attitude of the second message is “I have a problem. Could you give me a hand?”

### 3.1.1 Designing a Good User Interface

Designing a good user interface is an iterative process. First, you design and implement a user interface using appropriate techniques. Then you evaluate the design. The results of the evaluation feed the next design and implementation. You stop the process when you have met your design goals or you run out of time and/or money.

Note that if you have different user communities (or the same user with different jobs), you may need different user interfaces, customisable user interfaces or both. For example, Microsoft Word provides four user interfaces: normal, outline, page layout and master. In addition, Microsoft Word provides a host of customisation features for the keyboard, menu and toolbars.

While design is important, the real key to creating a good user interface is in your evaluation techniques. Obviously, you should use your own user interface. If you can't use it, how can anyone else? Next, get feedback from your testers.

The best evaluations are done by watching over the shoulder of the user. The key here is watching. If you are telling the user what to do, you will never find out if your interface is easy to use. Let the user figure it out himself or herself. If the user has to ask you what to do or pauses to figure out what to do next, you may need to work on your interface. If

the user grimaces, find out why. Learn from the experience. Some of the most innovative designs were shot down when the users could not figure them out.

You will need both new and experienced users for testing your interface. The new users will help you determine if you meet the Zero Training goal. The experienced users will let you know if your methods for meeting the Zero Training goal interfere with getting work done once the user has learned the software.

### 3.1.2 Tips for Designing a Good User Interface

- **Consistency:** The most important thing that you can possibly do is make sure that your user interface works consistently. If you can double-click on items in one list and have something happen, then you should be able to double-click on items in any other list and have the same sort of thing happen. Put your buttons in consistent places on all of your windows, use the same wording in labels and messages, and use a consistent colour scheme throughout. Consistency in your user interface allows your users to build an accurate mental model of the way that it works, and accurate mental models lead to lower training and support costs.
- **Set standards and stick to them:** The only way that you'll be able to ensure consistency within your application is to set design standards and then stick to them. The best approach is to adopt an industry standard and then fill any missing guidelines that are specific to your needs. Industry standards, such as the ones set by IBM (1993) and Microsoft (1995), will often define 95-99% of what you need. By adopting industry standards, you are not only taking advantage of the work of others, you are also increasing the chances that your application will look and feel like other applications that your users purchase or have built. User interface design standards should be set during the define infrastructure stage.
- **Explain the rules:** Your users need to know how to work with the application that you built for them. When an application works consistently, it means you only have to explain the rules once. This is a lot easier than explaining in detail exactly how to use each and every feature in an application step by step.
- **Support both novices and experts:** Although a library-catalog metaphor might be appropriate for casual users of a library system, library patrons, it probably is not all that effective for expert users, librarians. Librarians are highly trained people who

are able to use complex search systems to find information in a library; therefore you should consider building a set of search screens to support their unique needs.

- **Navigation between screens is important:** If it is difficult to get from one screen to another, then your users will quickly become frustrated and give up. When the flow between screens matches the flow of the work that the user is trying to accomplish, then your application will make sense to your users. Because different users work in different ways, your system will need to be flexible enough to support their various approaches. Interface-flow diagrams can be used during the model stage to model the flow between screens.
- **Navigation within a screen is important:** In Western societies, people read left to right and top to bottom. Because people are used to this, so you should design screens that are also organised left to right and top to bottom. All you want is to organise navigation between widgets on your screen in a manner that users will find familiar to them.
- **Word your messages and labels appropriately:** The text that you display on your screens is a primary source of information for your users. If your text is worded poorly then your interface will be perceived poorly by your users. Using full words and sentences, as opposed to abbreviations and codes makes your text easier to understand. Your messages should be worded positively, imply that the user is in control, and provide insight into how to use the application properly. For example, which message do you find more appealing “You have input the wrong information” or “An account number should be 8 digits in length?” Furthermore, your messages should be worded consistently and displayed in a consistent place on the screen. Although the messages “The person’s first name must be input.” and “An account number should be input.” are separately worded well, together they are inconsistent. In light of the first message, a better wording of the second message would be “The account number must be input” to make the two messages consistent.
- **Understand your widgets:** You should use the right widget (widgets are interface elements that the users interact with) for the right task, helping to increase the consistency in your application and probably making it easier to build the application in the first place. The only way that you can learn how to use widgets properly is to read and understand the user-interface standards and guidelines that your organisation has adopted.

- **Look at other applications with a grain of salt:** Unless you know that another application follows the user-interface standards and guidelines of your organisation, you must not assume that the application is doing things right. Although it is always a good idea to look at the work of others to get ideas, until you know how to distinguish between good and bad user-interface design you have to be careful. Too many developers make the mistake of imitating the user interface of another application that was poorly designed.
- **Use colour appropriately:** Colour should be used sparingly in your applications, and if you do use it you must also use a secondary indicator. The problem is that some of your users may be colour blind – if you are using colour to highlight something on a screen, then you need to do something else to make it stand out if you want people to notice it, such as display a symbol beside it. You also want to use colours in your application consistently so that you have a common look and feel throughout your application. Also, colour generally does not port well between platforms – what looks good on one system often looks poor on another system. We have all been to presentations where the presenter said “it looks good on my machine at home.”
- **Follow the contrast rule:** If you are going to use colour in your application, you need to ensure that your screens are still readable. The best way to do this is to follow the contrast rule: Use dark text on light backgrounds and light text on dark backgrounds. It is very easy to read blue text on a white background but very difficult to read blue text on a red background. The problem is that there is not enough contrast between blue and red to make it easy to read, whereas there is a lot of contrast between blue and white.
- **Use fonts appropriately:** Old English fonts might look good on the covers of William Shakespeare’s plays, but they are really difficult to read on a screen. Use fonts that are easy to read, such as serif fonts, Times Roman. Furthermore, use your fonts consistently and sparingly. A screen using two or three fonts effectively looks a lot better than a screen that uses five or six. Never forget that you are using a different font every time you change the size, style (bold, italics, underlining,), typeface, or colour.
- **Grey things out, do not remove them:** You often find that at certain times it is not applicable to give your users access to all the functionality of an application. You need to select an object

before you can delete it, so to reinforce your mental model the application should do something with the Delete button and/or menu item. Should the button be removed or greyed out? Grey it out, never remove it. By greying things out when they shouldn't be, users can start building an accurate mental model as to how your application works. If you simply remove a widget or menu item instead of greying it out then it is much more difficult for your users to build an accurate mental model, because they only know what is currently available and not what is not available to them. The old adage that out of sight is out of mind is directly applicable here.

- **Use non destructive default buttons:** It is quite common to define a default button on every screen, the button that gets invoked if the user presses the return/enter key. The problem is that sometimes people will accidentally hit the enter/return key when they do not mean to, consequently invoking the default button. Your default button shouldn't be something that is potentially destructive, such as delete or save (perhaps your user really did not want to save the object at that moment).
- **Alignment of fields:** When a screen has more than one editing field, you need to organise the fields in a way that is both visually appealing and efficient. As shown in figure 1.I, the best way to do so is to left-justify edit fields, or in other words make the left-hand side of each edit field line up in a straight line, one over the other. The corresponding labels should be right justified and placed immediately beside the field. This is a clean and efficient way to organise the fields on a screen.
- **Justify data appropriately:** For columns of data it is common practice to right justify integers, decimal align floating point numbers, and left justify strings.
- **Do not create busy screens:** Crowded screens are difficult to understand and hence are difficult to use. Experimental results show that the overall density of the screen should not exceed 40%, whereas local density within groupings shouldn't exceed 62%.
- **Group things on the screen effectively:** Items that are logically connected should be grouped together on the screen to communicate that they are connected, whereas items that have nothing to do with each other should be separated. You can use whitespace between collections of items to group them and/or you can put boxes around them to accomplish the same thing.

- **Open windows in the centre of the action:** When your user double-clicks on an object to display its edit/detail screen then his or her attention is on that spot. Therefore it makes sense to open the window in that spot, not somewhere else.
- **Pop-up menus should not be the only source of functionality:** Your users cannot learn how to use your application if you hide major functionality from them. One of the most frustrating practices of developers is to misuse pop-up, also called context-sensitive, menus. Typically there is a way to use the mouse on your computer to display a hidden pop-up menu that provides access to functionality that is specific to the area of the screen that you are currently working on.

**Poor Alignment**

Name:

Date of Birth:

ID.....:

**Good Alignment**

Name:

Date of Birth:

ID:

**Fig. 1.1: Showing that alignment of fields is critical**

### 3.2 Understanding Users

You must understand the user to be able to put a happy face on your application. You should understand the user's job, how the software fits in with that job and how the user goes about getting the job done. You need to approach the design of software from the user's viewpoint not from an abstract requirements document. Specifically, you should understand what the user will be doing with the application. If you can think like a user, you can create a much better user interface.

Here are some basic principles to remember about users:

- Your software is like a hammer - The user doesn't really care how well crafted it is, the user just wants nails put in the wall. Users just want to do their job (or play their game). They don't



care about you or your software. Your software is just an expedient tool to take the user where the user wants to go.

- Given a selection of hammers to buy at the hardware store - The user will select the one which will be most fun to use. Of course, this varies by user - some will want the plastic handle, some the wood, some the green, etc. When evaluating your software, users are often swayed by looks, not function. Thus, steps taken to make the product look good (nice icons, pictures, good colour scheme, fields aligned, etc.) will often favourably enhance evaluations of your software.
- It had better drive nails - The user will not really know if your software is adequate to the job until the user has used the software to do the actual work. From an interface perspective, the software should not look like it can do more than it can.
- Some users will try to use a hammer to drive a screw - If your software is good, some user somewhere will try to use the software for some purpose for which you never intended it to be used. Obviously, you cannot design a user interface to deal with uses you cannot foresee. There is no single rigid model of the right way to use the software, so build in flexibility.
- Users will not read an instruction manual for a hammer- They won't read one for your software either, unless they really have to. Users find reading instruction manuals almost as unpleasurable as dental work.
- A user reading the instruction manual for a hammer is in serious trouble - When you create your help system (and printed manual), remember that the user will only resort to those materials if he or she is in trouble. The user will want a problem solved as fast and as easily as possible.
- Hammers don't complain - You should try to eliminate error messages and any error messages your program needs should have the right attitude.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the essentials of good interface design. You have also learnt the necessary tips needed for designing a good interface and the need for understanding various users.

## 5.0 SUMMARY

In this unit, we have learnt:

- the essentials of interface design with emphasis on the features, functions and the face of the software
- that designing a good user interface which has been described as an iterative process involves designing, implementing, evaluating and redesigning until all removable errors have been taken care of
- the tips necessary for designing a good user interface which includes consistency, setting standards and sticking to them, supporting of both novices and experts, e.t.c.
- understanding the user's job, how the software fits in with that job and how the user goes about getting the job done.

### SELF- ASSESSMENT EXERCISE

- i. How do you ensure that the interface supports both novices and experts?
- ii. Write short notes on any three tips necessary for designing a good user interface.

## 6.0 TUTOR- MARKED ASSIGNMENT

- i. How do you ensure that the interface support both novices and expert?
- ii. Write short note on the design of a user interface for a user with hearing disability.

## 7.0 REFERENCES/FURTHER READING

Wikipedia.org

[www://www.linfo.org/gui.html](http://www.linfo.org/gui.html)

## UNIT 3 GRAPHICAL USER INTERFACE

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Graphical User Interface
  - 3.2 History of Graphical User Interface
  - 3.3 Elements of Graphical User Interface
  - 3.4 Three Dimensional Graphical User Interface
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the general concept of Graphical User Interface (GUI) and also the history and elements of graphical user interface. The concept of three dimensional (3D) graphical user interface is also introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe a graphical user interface
- explain the history behind graphical user interface
- list the elements of a graphical user interface
- describe the three-dimensional user interfaces.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Graphical User Interface

Graphical User Interfaces, also known as GUIs, offer a consistent visual language to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. This explains the most common elements of the visual language interfaces.

A **graphical user interface** is a type of user interface which allows people to interact with electronic devices such as computers; hand-held devices such as MP3 Players, Portable Media Players or Gaming

devices; household appliances and office equipment with images rather than text commands. A *GUI* offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

The term *GUI* is historically restricted to the scope of two-dimensional display screens with display resolutions capable of describing generic information, in the tradition of the computer science research at Palo Alto Research Centre (PARC). The term *GUI* earlier might have been applicable to other high-resolution types of interfaces that are non-generic, such as videogames, or not restricted to flat screens, like volumetric displays.

## 3.2 History of Graphical User Interface

### Precursor to GUI

The precursor to GUIs was invented by researchers at the Stanford Research Institute, led by Douglas Engelbart. They developed the use of text-based hyperlinks manipulated with a mouse for the On-Line System. The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC, who went beyond text-based hyperlinks and used a GUI as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. As a result, some people call this class of interface PARC User Interface (PUI) (note that PUI is also an acronym for Perceptual User Interface). Ivan Sutherland developed a pointer-based system called the Sketchpad in 1963. It used a light-pen to guide the creation and manipulation of objects in engineering drawings.

### PARC User Interface

The PARC User Interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC User Interface employs a pointing device in addition to a keyboard. These aspects can be emphasised by using the alternative acronym WIMP, which stands for **W**indows, **I**cons, **M**enus and **P**ointing device.

### Evolution

Following PARC, the first GUI-centric computer operating model was the Xerox 8010 Star Information System in 1981 followed by the Apple Lisa (which presented concept of menu bar as well as window controls) in 1982 and the Atari ST and Commodore Amiga in 1985.

The GUIs that are familiar to most people today are Microsoft Windows, Finder Interface (Mac OS X), and the X Window System interfaces. Apple, IBM and Microsoft used many of Xerox's ideas to develop products, and IBM's Common User Access specifications formed the basis of the user interface found in Microsoft Windows, IBM OS/2 Presentation Manager, and the Unix Motif toolkit and window manager. These ideas evolved to create the interface found in current versions of Microsoft Windows, as well as in Mac OS X and various desktop environments for Unix-like operating systems, such as Linux. Thus most current GUIs have largely common idioms.

### **Post-WIMP Interfaces**

Smaller mobile devices such as PDAs and smart phones typically use the WIMP elements with different unifying metaphors, due to constraints in space and available input devices. Applications for which WIMP is not well suited may use newer interaction techniques, collectively named as post-WIMP user interfaces.

Some touch-screen-based operating systems such as Apple's iPhone OS currently use post-WIMP styles of interaction. The iPhone's use of more than one finger in contact with the screen allows actions such as pinching and rotating, which are not supported by a single pointer and mouse.

A class of GUIs sometimes referred to as post-WIMP include 3D compositing window manager such as Compiz, Desktop Window Manager, and LG3D. Some post-WIMP interfaces may be better suited for applications which model immersive 3D environments, such as Google Earth.

## **3.3 Elements of Graphical User Interfaces**

A GUI uses a combination of technologies and devices to provide a platform which the user can interact with in order to achieve the tasks of gathering and producing information.

Series of elements conforming to visual languages have evolved to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. The most common combination of such elements in GUIs is the WIMP paradigm, especially in personal computers.

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of

the static elements on which the user can interact, and define the appearance of the interface.

The key elements of GUI are divided into two categories viz Structural and Interactive elements.

### 3.3.1 Structural Elements

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of the static elements on which the user can interact, and define the appearance of the interface.

#### Window

A window is an area on the screen that displays information, with its contents being displayed independently from the rest of the screen. An example of a window is what appears on the screen when the "My Documents" icon is clicked in the Windows Operating System. It is easy for a user to manipulate a window: it can be opened and closed by clicking on an icon or application, and it can be moved to any area by dragging it (that is, by clicking in a certain area of the window – usually the title bar along the top – and keeping the pointing device's button pressed, then moving the pointing device). A window can be placed in front or behind another window, its size can be adjusted, and scrollbars can be used to navigate the sections within it. Multiple windows can also be opened at one time, in which case each window can display a different application or file – this is very useful when working in a multitasking environment. The system's memory is the only limitation to the number of windows that can be opened at once. There are also many types of specialised windows.

- A **Container Window**: a window that is opened while invoking the icon of a mass storage device, directory or folder and which is presenting an ordered list of other icons that could be again some other directories, or data files or may be even executable programs. All modern container windows could present their content on screen either by acting as browser windows or text windows. Their behaviour can automatically change according to the choices of the single users and their preferred approach to the graphical user interface.
- A **browser window**: allows the user to move forward and backward through a sequence of documents or web pages. Web browsers are examples of these types of windows.
- **Text terminal windows**: are designed for embedding interaction with text user interfaces within the overall graphical interface.

MS-DOS and UNIX consoles are examples of these types of windows.

- A **child window**: opens automatically or as a result of a user activity in a parent window (A parent window can be any type of window). Pop-up windows on the Internet can be child windows.
- A **message window**, or **dialog box**: is a type of child window. These are usually small and basic windows that are opened by a program to display information to the user and/or get information from the user. They usually have a button that must be pushed before the program can be resumed.

## Menus

Menus allow the user to execute commands by selecting from a list of choices. Options are selected with a mouse or other pointing device within a GUI. A keyboard may also be used. Menus are convenient because they show what commands are available within the software. This limits the amount of documentation the user reads to understand the software.

- A **menu bar** is displayed horizontally across the top of the screen and/or along the top of some or all windows. A pull-down menu is commonly associated with this menu type. When a user clicks on a menu option, the pull-down menu will appear.
- A **menu** has a visible title within the menu bar. Its contents are only revealed when the user selects it with a pointer. The user is then able to select the items within the pull-down menu. When the user clicks elsewhere, the contents of the menu will disappear.
- A **context menu** is invisible until the user performs a specific mouse action, like pressing the right mouse button. When the software-specific mouse action occurs the menu will appear under the cursor.
- **Menu extras** are individual items within or at the side of a menu.

## Icons

An icon is a small picture that represents objects such as a file, program, web page, or command. Icons are used as a quick way to execute commands, open documents, and run programs. They are also very useful when searching for an object in a browser list, because in many operating systems all documents using the same extension will have the same icon.

## Controls (or Widgets)

The interface element that a computer user interacts with is known as a **control** or **widget**.

### *Window*

A paper-like rectangle that represents a "window" into a document, form, or design area.

### *Pointer (or mouse cursor)*

The spot where the mouse "cursor" is currently referencing.

### *Text box*

A box in which texts or numbers are entered.

### *Button*

An equivalent to a push-button as found on mechanical or electronic instruments.

### *Hyperlink*

Text with some kind of indicators (usually underlining and/or colour) that shows that clicking it will take one to another screen or page.

### *Drop-down list*

A list of items from which to select: The list normally only displays items when a special button or indicator is clicked.

### *Check box*

A box which indicates an "on" or "off" state via a check-mark or an "×".

### *Radio button*

A button, similar to a check-box, except that only one item in a group can be selected. Its name comes from the mechanical push-button group on a car radio receiver. Selecting a new item from the group's buttons also deselects the previously selected button.

### *Data grid*

A spreadsheet-like grid that allows numbers or text to be entered in rows and columns.

## **Tabs**

A tab is typically a rectangular small box which usually contains a text label or graphical icon associated with a view pane. When activated the view pane, or window, displays widgets associated with that tab; groups of tabs allow the user to switch quickly between different widgets. This



is used in the web browsers Firefox, Internet Explorer, Konqueror, Opera, and Safari. With these browsers, you can have multiple web pages open at once in one window, and quickly navigate between them by clicking on the tabs associated with the pages. Tabs are usually placed in groups at the top of a window, but may also be grouped on the side or bottom of a window.

### **3.3.2 Interaction Elements**

Some common idioms for interaction have evolved in the visual language used in GUIs. Interaction elements are interface objects that represent the state of an ongoing operation or transformation, either as visual reminders of the user intent (such as the pointer), or as affordances showing places where the user may interact.

#### **Cursor**

A cursor is an indicator used to show the position on a computer monitor or other display devices that will respond to inputs from a text input or pointing devices.

#### **Pointer**

One of the most common components of a GUI on the personal computer is a pointer: a graphical image on a screen that indicates the location of a pointing device, and can be used to select and move objects or commands on the screen. A pointer commonly appears as an angled arrow, but it can vary within different programs or operating systems. Example of this can be found within text-processing applications, which use an I-beam pointer that is shaped like a capital I, or in web browsers, which often indicate that the pointer is over a hyperlink by turning the pointer in the shape of a gloved hand with outstretched index finger.

The use of a pointer is employed when the input method, or pointing device, is a device that can move fluidly across a screen and select or highlight objects on the screen. Pointer trails can be used to enhance its visibility during movement. In GUIs where the input method relies on hard keys, such as the five-way key on many mobile phones, there is no pointer employed, and instead the GUI relies on a clear focus state.

#### **Selection**

A selection is a list of items on which user operations will take place. The user typically adds items to the list manually, although the computer may create a selection automatically.

## Adjustment handle

A handle is an indicator of a starting point for a drag and drop operation. Usually the pointer shape changes when placed on the handle, showing an icon that represents the supported drag operation.

## SELF-ASSESSMENT EXERCISE 1

Identify and study these elements within Window operating system

### 3.4 Three-dimensional User Interfaces

For typical computer displays, *three-dimensional* are a misnomer—their displays are two-dimensional. Three-dimensional images are projected on them in two dimensions. Since this technique has been in use for many years, the recent use of the term three-dimensional must be considered a declaration by equipment marketers that the speed of three dimensions to two dimensions projection is adequate to use in standard GUIs.

#### Motivation

Three-dimensional GUIs are quite common in science fiction literature and movies, such as in *Jurassic Park*, which features Silicon Graphics' three-dimensional file manager, "File system navigator", an actual file manager that never got much widespread use as the user interface for a Unix computer. In fiction, three-dimensional user interfaces are often immersible environments like William Gibson's *Cyberspace* or Neal Stephenson's *Metaverse*.

Three-dimensional graphics are currently mostly used in computer games, art and computer-aided design (CAD). There have been several attempts at making three-dimensional desktop environments like Sun's Project Looking Glass or SphereXP from Sphere Inc. A three-dimensional computing environment could possibly be used for collaborative work. For example, scientists could study three-dimensional models of molecules in a virtual reality environment, or engineers could work on assembling a three-dimensional model of an airplane. This is a goal of the Croquet project and Project Looking Glass.

#### Technologies

The use of three-dimensional graphics has become increasingly common in mainstream operating systems, from creating attractive interfaces—eye candy—to functional purposes only possible using three dimensions. For example, user switching is represented by rotating

a cube whose faces are each user's workspace, and window management is represented in the form or via a Rolodex-style flipping mechanism in Windows Vista (see Windows Flip 3D). In both cases, the operating system transforms windows on-the-fly while continuing to update the content of those windows.

Interfaces for the X Window System have also implemented advanced three-dimensional user interfaces through compositing window managers such as Beryl, Compiz and KWin using the AIGLX or XGL architectures, allowing for the usage of OpenGL to animate the user's interactions with the desktop.

Another branch in the three-dimensional desktop environment is the three-dimensional GUIs that take the desktop metaphor a step further, like the BumpTop, where a user can manipulate documents and windows as if they were "real world" documents, with realistic movement and physics.

The Zooming User Interface (ZUI) is a related technology that promises to deliver the representation benefits of 3D environments without their usability drawbacks of orientation problems and hidden objects. It is a logical advancement on the GUI, blending some three-dimensional movement with two-dimensional or "2.5D" vector objects.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to graphical user interface (GUI). The history of graphical user interface was also discussed. The elements of graphical user interface were also explained. You were also introduced to three-dimensional user interfaces.

#### **5.0 SUMMARY**

In this unit, you have learnt the:

- introduction to graphical user interface which is a type of user interface that allows people to interact with electronic devices such as computers, hand-held devices, household appliances and office equipment with images rather than text commands
- history of graphical user interface, precursor to GUI, PARC user interface and the evolution of other graphical user interfaces
- elements of graphical user interface which are divided into two categories that includes structural and interactive elements.

## **SELF-ASSESSMENT EXERCISE 2**

- i. What do you understand by graphical user interface?
- ii. Explain the structural and iterative elements of graphical user interface.

## **6.0 TUTOR- MARKED ASSIGNMENT**

Explain the PARC graphical user interface

## **7.0 REFERENCES/FURTHER READING**

<http://www.linfo.org/gui.html>. Retrieved on 12 November 2008.

[www.wikipedia.com](http://www.wikipedia.com)

## UNIT 4 HUMAN- COMPUTER INTERACTION

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Human -Computer Interaction (HCI)
  - 3.2 Goals of HCI
  - 3.3 Differences with Other Related Fields
  - 3.4 Future Development of HCI
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor -Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In this unit, you will be introduced to Human-Computer Interaction (HCI) and its differences with other related fields. The goals and future development of human- computer interaction and the general concept of human-computer interface will be introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the term human- computer interaction
- identify the various goals of human -computer interaction
- differentiate human -computer interaction from other related fields
- describe the future development of HCI and explain the human - computer interface.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Human -Computer Interaction

**Human–Computer Interaction (HCI)** is the study of interaction between people (users) and computers. It involves the study intersection of computer science, behavioural sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply *interface*), which includes both software and hardware. The association for computing machinery defines *human-computer interaction* as "a discipline concerned with the design,

evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them” Since human-computer interaction studies a human and a machine together, it draws its supporting knowledge from the machine and human sides. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, statistics, linguistics, social sciences, cognitive psychology, and human performance are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success. HCI is sometimes referred to as **Man–Machine Interaction (MMI)** or **Computer–Human Interaction (CHI)**.

### 3.1.1 Human–Computer Interface

The human–computer interface can be described as the point of communication between the human user and the computer. The flow of information between the human and computer is defined as the loop of interaction. The loop of interaction has several aspects to it including:

- **Task Environment:** The conditions and goals set upon the user.
- **Machine Environment:** The environment that the computer is connected to, i.e. a laptop in a college student's dorm room.
- **Areas of the Interface:** Non-overlapping areas involve processes of the human and computer not pertaining to their interaction. Meanwhile, the overlapping areas only concern themselves with the processes pertaining to their interaction.
- **Input Flow:** Begins in the task environment as the user has some task that requires using their computer.
- **Output:** The flow of information that originates in the machine environment.
- **Feedback:** Loops through the interface that evaluate, moderate, and confirm processes as they pass from the human through the interface to the computer and back.

### 3.2 Goals of HCI

The basic goal of HCI is to improve the interactions between users and computers by making computers more usable and responsive to the user's needs. Specifically, HCI is concerned with:

- methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface

within given constraints, optimising for a desired property such as learning ability or efficiency of use)

- methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- techniques for evaluating and comparing interfaces
- developing new interfaces and interaction techniques
- developing descriptive and predictive models and theories of interaction.

A long term goal of HCI is to design systems that minimise the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

Researchers in HCI are interested in developing new design methodologies, experimenting with new hardware devices, prototyping new software systems, exploring new paradigms for interaction, and developing models and theories of interaction.

### **3.3 Differences with Related Fields**

HCI focuses on user interface design mainly for users of computer system and promotes effective interaction between computers and users (human). User Interface Design is concerned with the users of devices such as computers, appliances, machines, mobile communication devices, software applications, and websites. In HCI, efficient user interface is critical.

HCI differs from human factors in that the focus is more on users working specifically with computers, rather than other kinds of machines or designed artifacts. There is also a focus in HCI on how to implement the computer software and hardware mechanisms to support human-computer interaction. Thus, human factors are a broader term; HCI could be described as the human factors of computers, although some experts try to differentiate these areas.

According to some experts, HCI also differs from ergonomics (which will be fully introduced in the next unit) in that there is less focus on repetitive work-oriented tasks and procedures, and much less emphasis on physical stress and the physical form or industrial design of the user interface, such as keyboards and mice. However, this does not take full account of ergonomics, which recently has gained a much broader focus (equivalent to human factors). Cognitive ergonomics, for example, is a

part of ergonomics, of which *software ergonomics* (an older term, essentially the same as HCI) is a part.

Three areas of study have substantial overlap with HCI even as the focus of inquiry shifts. In the study of Personal Information Management (PIM), human interactions with the computer are placed in a larger informational context - people may work with many forms of information, some computer-based, many not (e.g., whiteboards, notebooks, sticky notes, refrigerator magnets) in order to understand and effect desired changes in their world. In Computer Supported Cooperative Work (CSCW), emphasis is placed on the use of computing systems in support of the collaborative work of a group of people. The principles of Human Interaction Management (HIM) extend the scope of CSCW to an organisational level which can be implemented without the use of computer systems.

### 3.4 Future Development of HCI

As the means by which humans interact with computers continues to evolve rapidly, human-computer interaction is affected by the forces shaping the nature of future computing. These forces include:

- decreasing hardware costs leading to larger memories and faster systems
- miniaturisation of hardware leading to portability
- reduction in power requirements leading to portability
- new display technologies leading to the packaging of computational devices in new forms
- specialised hardware leading to new functions
- increased development of network communication and distributed computing
- increasingly widespread use of computers, especially by people who are outside the computing profession
- increasing innovation in input techniques (i.e., voice, gesture, pen), combined with lowering cost, leading to rapid computerisation by people previously left out of the "computer revolution"
- wider social concerns leading to improved access to computers by currently disadvantaged groups.

The future for HCI is expected to include the following features:

**Ubiquitous communication:** Computers will communicate through high speed local networks, nationally over wide-area networks, and portably via infrared, ultrasonic, cellular, and other technologies. Data



and computational services will be portably accessible from many if not most locations to which a user travels.

**High functionality systems:** Systems will have large numbers of functions associated with them. There will be so many systems that most users, technical or non-technical, will not have time to learn them in the traditional way (e.g., through thick manuals).

**Mass availability of computer graphics:** Computer graphics capabilities such as image processing, graphics transformations, rendering, and interactive animation will become widespread as inexpensive chips become available for inclusion in general workstations.

**Mixed media:** Systems that will handle images, voice, sounds, video, text, formatted data. These will be exchangeable over communication links among users. The separate worlds of consumer electronics (e.g., stereo sets, VCRs, televisions) and computers will partially merge. Computer and print worlds will continue to cross assimilate each other.

**High-bandwidth interaction:** The rate at which humans and machines interact will increase substantially due to the changes in speed, computer graphics, new media, and new input/output devices. This will lead to some qualitatively different interfaces, such as virtual reality or computational video.

**Large and thin displays:** New display technologies will finally mature enabling very large displays and also displays that are thin, light weight, and have low power consumption. This will have large effects on portability and will enable the development of paper-like, pen-based computer interaction systems very different in feel from desktop workstations of the present.

**Embedded computation:** Computation will pass beyond desktop computers into every object for which uses can be found. The environment will be alive with little computations from computerised cooking appliances. to lighting and plumbing fixtures to window blinds to automobile braking systems to greeting cards. To some extent, this development is already taking place. The difference in the future is the addition of networked communications that will allow many of these embedded computations to coordinate with each other and with the user. Human interfaces to these embedded devices will in many cases be very different from those appropriate to workstations.

**Augmented reality:** A common staple of science fiction, augmented reality refers to the notion of layering relevant information into our

vision of the world. Existing projects show real-time statistics to users performing difficult tasks, such as manufacturing. Future work might include augmenting our social interactions by providing additional information about those we converse with.

**Group interfaces:** Interfaces to allow groups of people to coordinate will be common (e.g., for meetings, for engineering projects, for authoring joint documents). These will have major impacts on the nature of organisations and on the division of labour. Models of the group design process will be embedded in systems and will cause increased rationalisation of the design.

**User tailorability:** Ordinary users will routinely tailor applications to their own use and will use this power to invent new applications based on their understanding of their own domains. Users, with their deeper knowledge of their own knowledge domains, will increasingly be important sources of new applications at the expense of generic systems programmers (with systems expertise but low domain expertise).

**Information utilities:** Public information utilities (such as home banking and shopping) and specialised industry services (e.g., weather for pilots) will continue to proliferate. The rate of proliferation will accelerate with the introduction of high-bandwidth interaction and the improvement in quality of interfaces.

## 4.0 CONCLUSION

In this unit, you have been introduced to the concepts of human - computer interaction. You have also been introduced to the various goals of HCI and also the difference between HCI and other related fields. The future of HCI was also discussed. The concept of the human-computer interface was also introduced.

## 5.0 SUMMARY

In this unit, you have learnt the following:

- introduction to human-computer interaction (HCI) which is the study of interaction between people (users) and computers. HCI is also sometimes referred to as man-machine interaction (MMI) or computer-human interaction (CHI)
- the human-computer interface which can be described as the point of communication between the human user and the computer

- the goal of HCI which is basically to improve the interactions between users and computers by making computers more usable and receptive to the user's needs
- the difference between HCI and other related fields like graphical user interface, ergonomics e.t.c.
- the future development in HCI like Ubiquitous communication, High functionality systems, Mass availability of computer graphics e.t.c.

### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by Human -Computer Interaction?
- ii. Describe the Human -Computer Interface

### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly the future development in HCI.

### **7.0 REFERENCES/FURTHER READING**

More discussion of the differences between these terms can be found in the ACM SIGCHI Curricula for Human-Computer Interaction

Brown, C. M. (1998). *Human-Computer Interface Design Guidelines*. Intellect Books. pp.2–3.

Green, P. (2008). Iterative Design. Lecture presented in Industrial and Operations Engineering 436 Human Factors in Computer Systems, University of Michigan, Ann Arbor, MI, February 4, 2008.

Wickens, C. D., John, D. L, Yili, L., & Sallie, E. G. (2004). *An Introduction to Human Factors Engineering*. (2nd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. pp.185–193.

## **UNIT 5     ERGONOMICS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Ergonomics
  - 3.2 Five Aspects of Ergonomics
  - 3.3 History of Ergonomics
  - 3.4 Ergonomics in Workplace
  - 3.5 Efficiency and Ergonomics
  - 3.6 Benefits of Ergonomics
  - 3.7 Fields of Ergonomics
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

This unit describes in great detail the various aspects of ergonomics alongside the history behind it. The efficiency and benefits of ergonomics will also be discussed. Different fields of ergonomics will also be highlighted.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- define the term ergonomics
- identify the various aspects of ergonomics
- explain the history of ergonomics
- describe efficiency and ergonomics
- identify the various benefits of ergonomics
- list the various fields of ergonomics.

### **3.0 MAIN CONTENT**

#### **3.1 Introduction to Ergonomics**

Ergonomics is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work.

The International Ergonomics Association has adopted this technical definition: ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimise human well-being and overall system performance.

Ergonomics is the science of making things comfy. It also makes things efficient. However for simplicity, ergonomics makes things comfortable and efficient.

At its simplest definition, ergonomics literally means the science of work. So ergonomists, i.e. the practitioners of ergonomics, study work, how work is done and how to work better. It is the attempt to make work better that ergonomics becomes so useful.

However, what you, or the user, is most concerned with is, “How can I use the product or service, will it meet my needs, and will I like using it?” Ergonomics helps define how it is used, how it meets you needs, and most importantly if you like it. It makes things comfy and efficient.

Ergonomics is concerned with the ‘fit’ between people and their work. It takes account of the worker's capabilities and limitations in seeking to ensure that tasks, equipment, information and the environment suit each worker.

To assess the fit between a person and his/her work, ergonomists consider the job being done and the demands on the worker; the equipment used (its size, shape, and how appropriate it is for the task), and the information used (how it is presented, accessed, and changed). Ergonomics draws on many disciplines in its study of humans and their environments, including anthropometry, biomechanics, mechanical engineering, industrial engineering, industrial design, kinesiology, physiology and psychology.

### **3.2 Five Aspects of Ergonomics**

There are five aspects of ergonomics: safety, comfort, ease of use, productivity/performance, and aesthetics. From these aspects of ergonomics, examples are given of how products or systems could benefit from redesign based on ergonomic principles.

- Safety – This has to do with the ability to use a device or work with a device without short or long term damage to parts of the body. For example in Medicine bottles: The print on them could be larger so that a sick person who may have bad vision (due to

sinuses, etc.) can easily see the dosages and label. Ergonomics could design the print style, colour and size for optimal viewing.

- Comfort – Comfort in the human-machine interface is usually noticed first. Physical comfort in how an item feels is pleasing to the user. If you do not like to touch it you won't. If you do not touch it you will not operate it. If you do not operate it, then it is useless. For example, in Alarm clock display: Some displays are harshly bright, drawing one's eye to the light when surroundings are dark. Ergonomic principles could re-design this based on contrast principles.
- Ease of use – This has to do with the ability to use a device with no stress. For example in Street Signs: In a strange area, many times it is difficult to spot street signs. This could be addressed with the principles of visual detection in ergonomics.
- Productivity/performance – For example in High-definition Television (HD TV): The sound on HD TV is much lower than regular TV. So when you switch from HD to regular, the volume increases dramatically. Ergonomics recognises that this difference in decibel level creates a difference in loudness and hurts human ears and this could be solved by evening out the decibel levels.
- Aesthetics - the look and feel of the object, the user experience.

### 3.3 History of Ergonomics

The foundations of the science of ergonomics appear to have been laid within the context of the culture of Ancient Greece. A good deal of evidence indicates that Hellenic civilisation in the 5th century BC used ergonomic principles in the design of their tools, jobs, and workplaces.

The term ergonomics is derived from the Greek words *ergon* [work] and *nomos* [natural laws] and first entered the modern lexicon when Wojciech Jastrzębowski used the word in his 1857 article *Rys ergonomji czyli nauki o pracy, opartej na prawdach poczerpniętych z Nauki Przyrody* (The Outline of Ergonomics, i.e. Science of Work, Based on the Truths Taken from the Natural Science).

Later, in the 19th century, Frederick Winslow Taylor pioneered the "Scientific Management" method, which proposed a way to find the optimum method for carrying out a given task. Taylor found that he could, for example, triple the amount of coal that workers were shoveling by incrementally reducing the size and weight of coal shovels

until the fastest shoveling rate was reached. Frank and Lillian Gilbreth expanded Taylor's methods in the early 1900s to develop "Time and Motion Studies". They aimed at improving efficiency by eliminating unnecessary steps and actions. By applying this approach, the Gilbreths reduced the number of motions in bricklaying from 18 to 4.5, allowing bricklayers to increase their productivity from 120 to 350 bricks per hour.

World War II marked the development of new and complex machines and weaponry, and these made new demands on operators' cognition. The decision-making, attention, situational awareness and hand-eye coordination of the machine's operator became keys to the success or failure of a task. It was observed that fully functional aircraft, flown by the best-trained pilots, still crashed. In 1943, Alphonse Chapanis, a lieutenant in the U.S. Army, showed that this so-called "pilot error" could be greatly reduced when more logical and differentiable controls replaced confusing designs in airplane cockpits. The experience of the World War II promoted the study. Man-Machine interaction became a major focus when weapons meant to attack the enemies were malfunctioning – attacking friends instead of foes.

In the decades since the war, ergonomics has continued to flourish and diversify. The Space Age created new human factors issues such as weightlessness and extreme g-forces. How far could environments in space be tolerated, and what effects would they have on the mind and body? The dawn of the Information Age has resulted in the new ergonomics field of human-computer interaction (HCI). Likewise, the growing demand for and competition among consumer goods and electronics has resulted in more companies including human factors in product design.

At home, work, or play new problems and questions must be resolved constantly. People come in all different shapes and sizes, and with different capabilities and limitations in strength, speed, judgment, and skills. All of these factors need to be considered in the design function. To solve design problems, physiology and psychology must be included with an engineering approach.

### 3.4 Ergonomics in Workplace



**Fig. 5.1: Description of Workplace Environment**

(Source: [www.wikipedia.org](http://www.wikipedia.org))

Fundamentals for the Flexible Workplace variability and compatibility with desk components, that flex from individual work activities to team settings. Workstations provide supportive ergonomics for task-intensive environments.

Outside of the discipline itself, the term 'ergonomics' is generally used to refer to physical ergonomics as it relates to the workplace (example ergonomic chairs and keyboards). Ergonomics in the workplace has to do largely with the safety of employees, both in the long and short-term. Ergonomics can help reduce costs by improving safety. This would decrease the money paid out in workers' compensation.

Workplaces may either take the reactive or proactive approach when applying ergonomics practices. Reactive ergonomics is when something needs to be fixed, and corrective action is taken. Proactive ergonomics is the process of seeking areas that could be improved and fixing the issues before they become a large problem. Problems may be fixed through equipment design, task design, or environmental design. Equipment design changes the actual, physical devices used by people. Task design changes what people do with the equipment. Environmental design changes the environment in which people work, but not the physical equipment they use.

### 3.5 Efficiency and Ergonomics

Efficiency is quite simply making something easier to do. Several forms of efficiency are:



- Reducing the strength required makes a process more physically efficient.
- Reducing the number of steps in a task makes it quicker (i.e. efficient) to complete.
- Reducing the number of parts makes repairs more efficient.
- Reducing the amount of training needed, i.e. making it more intuitive, gives you a larger number of people who are qualified to perform the task. Imagine how in-efficient trash disposal would be if your teenage child wasn't capable of taking out the garbage. Have you tried an ergonomic trash bag?

Efficiency can be found almost everywhere. If something is easier to do you are more likely to do it. If you do it more, then it is more useful. Again, utility is the only true measure of the quality of a design. And if you willingly do something more often you have a greater chance of liking it. If you like doing it you will be more comfortable doing it. So the next time you hear the term ergonomics you will know what it means to you. And I hope that is a comforting thought.

Ergonomics can help you in many ways. Among other things, it can benefit your life, health, productivity and accuracy. One of the best benefits of ergonomics is saving time. We never seem to have enough of it as it is, so why not try to get a little more out of your day? Ergonomics is about making things more efficient. By increasing the efficiency of a tool or a task, you tend to shorten the length of time it takes to accomplish your goal.

### **3.6 Benefits of Ergonomics**

The three main benefits of ergonomics are:

#### **Slim Down the Task**

Have you ever wondered why some things are so convoluted, cumbersome and chaotic? And they take forever to complete. And most of what you do does not aid the outcome.

For example, think back to the last time you got hired for a job, bought a house or car, or did something else that required a ton of paperwork. How many different forms did you write the same information on? That was not very ergonomic.

You can almost always make a task a little leaner. But first you have to understand the task. Ergonomics requires that tasks are properly slimed down and steps involved in a task are well written out. This is why task

analysis is normally done in ergonomics (Task analysis is fully discussed in module 2, unit 2).

Once you have all the steps written out, you need to take a good look at them and identify areas that you can "ergonomise":

1. *Repetition* – Look for steps that are repeated and see if they are all necessary.
2. *Order* – See if you can re-order the steps to optimise your effort.
3. *Synergy* – Can you combine things or somehow get more bang for your buck?
4. *Value Added* – Look at every step and make sure it adds value to the outcome. If it doesn't, cut it.
5. *Necessity* – Make sure the quantity of the step is needed. Do you really need to brush your teeth with 57 strokes, or will 32 do?

### **Simplify the Task**

You can also save time by simplifying the task. This is not just about reducing the number of steps, but making those steps easier to perform. The less training and/or skill that is required for a task, the quicker the pace at which it tends to get finished.

This is a great ergonomic tip, especially when the task requires more than one person. If you are trying to get your kids to pick up their toys before they go to bed, you can save a lot of time by making it easier. That is what the toy chest is for. Instead of having different places for different things, they can just throw everything in one place.

### **Increase Body Mechanism**

Ergonomic can increase your body mechanics. A good ergonomic tool acts as extension of your body enhancing capabilities. Some tools make you more effective and faster at completing a task. (Try cutting a log without an axe and see how long it takes you.)

## **3.7 Fields of Ergonomics**

### **Engineering Psychology**

*Engineering psychology* is an interdisciplinary part of Ergonomics and studies the relationships of people to machines, with the intent of improving such relationships. This may involve redesigning equipment, changing the way people use machines, or changing the location in which the work takes place. Often, the work of an engineering

psychologist is described as making the relationship more "user-friendly."

Engineering Psychology is an applied field of psychology concerned with psychological factors in the design and use of equipment. Human factors are broader than engineering psychology, which is focused specifically on designing systems that accommodate the information-processing capabilities of the brain.

### **Macroergonomics**

Macroergonomics is an approach to ergonomics that emphasises a broad system view of design, examining organisational environments, culture, history, and work goals. It deals with the physical design of tools and the environment. It is the study of the society/technology interface and their consequences for relationships, processes, and institutions. It also deals with the optimisation of the designs of organisational and work systems through the consideration of personnel, technological, and environmental variables and their interactions. The goal of macroergonomics is a completely efficient work system at both the macro- and micro-ergonomic level which results in improved productivity, and employee satisfaction, health, safety, and commitment. It analyses the whole system, finds how each element should be placed in the system, and considers all aspects for a fully efficient system. A misplaced element in the system can lead to total failure.

### **Seating Ergonomics**

The best way to reduce pressure in the back is to be in a standing position. However, there are times when you need to sit. When sitting, the main part of the body weight is transferred to the seat. Some weight is also transferred to the floor, backrest, and armrests. Where the weight is transferred is the key to a good seat design. When the proper areas are not supported, sitting in a seat all day can put unwanted pressure on the back causing pain.

The lumbar (bottom five vertebrae in the spine) needs to be supported to decrease disc pressure. Providing both a seat back that inclines backwards and has a lumbar support is critical to prevent excessive low back pressures. The combination which minimises pressure on the lower back is having a backrest inclination of 120 degrees and a lumbar support of 5 cm. The 120 degrees inclination means the angle between the seat and the backrest should be 120 degrees. The lumbar support of 5 cm means the chair backrest supports the lumbar by sticking out 5 cm in the lower back area.

Another key to reducing lumbar disc pressure is the use of armrests. They help by putting the force of your body not entirely on the seat and backrest, but putting some of this pressure on the armrests. Armrest needs to be adjustable in height to assure that shoulders are not overstressed.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the fundamental concepts of Ergonomics. You have also learnt the different aspect of ergonomics and also the history of ergonomics. Ergonomics in workplace was also discussed alongside achieving efficiency in ergonomics. The various benefits of ergonomics were also discussed. The various fields of ergonomics were also briefly explained.

#### **5.0 SUMMARY**

In this unit you, have learnt the following:

- introduction to ergonomics which is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work
- highlighting of the various aspect of ergonomics like safety, comfort, ease of use e.t.c.
- the history of ergonomics whose foundations appears to have been laid within the context of the culture of Ancient Greece
- the discussion of ergonomics in workplace and achieving efficiency in ergonomics
- explanation of the various benefits of ergonomics which was discussed in greater detail
- the discussion of various fields of ergonomics like engineering psychology, Macroergonomics, Seating ergonomics.

#### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by ergonomics?
- ii. Highlight the various benefits of ergonomics, giving examples.

#### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly any two fields of ergonomics.

## 7.0 REFERENCES/FURTHER READING

Berkeley Lab. *Integrated Safety Management: Ergonomics*. Website. Retrieved 9 July 2008.

Berkeley Lab. *Today at Berkeley Lab: Ergonomic Tips for Computer Users*. Retrieved 8 January 2009.

Brookhuis, K., Hedge, A., Hendrick, H., Salas, E., & Stanton, N. (2005). *Handbook of Human Factors and Ergonomics Models*. Florida: CRC Press.

Wickens & Hollands (2006). *Engineering Psychology and Human Performance*.

[www.wikipedia.com](http://www.wikipedia.com)

## **MODULE 1 INTRODUCTION TO USER INTERFACE DESIGN**

Unit 1	Fundamentals of User Interface Design
Unit 2	Designing Good User Interfaces
Unit 3	Graphical User Interface (GUI)
Unit 4	Human-Computer Interaction
Unit 5	Ergonomics

### **UNIT 1 FUNDAMENTALS OF USER INTERFACE DESIGN**

#### **CONTENTS**

1.0	Introduction
2.0	Objectives
3.0	Main Content
3.1	User Interface
3.2	Significance of User Interface
3.3	Types of User Interfaces
3.4	History of User Interfaces
3.5	User Interface Modes and Modalities
3.6	Introduction to Usability
4.0	Conclusion
5.0	Summary
6.0	Tutor- Marked Assignment
7.0	References/Further Reading

#### **1.0 INTRODUCTION**

Having read through the course guide, you will have a general understanding of what this unit is about and how it fits into the course as a whole. This unit describes the general fundamentals of user interface design.

#### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain the term user interface design
- identify the significance of user interface
- explain the history behind user interfaces
- describe the modalities and modes of user interface.

## 3.0 MAIN CONTENT

### 3.1 User Interface

The **user interface** (also known as Human Machine Interface (HMI) or Man-Machine Interface (MMI)) is the aggregate of means by which people—the *users*—interact with the *system*—a particular machine, device, computer program or other complex tool.

**User interface** is the point at which a user or a user department or organisation interacts with a computer system. The part of an interactive computer program that sends messages to and receives instructions from a terminal user.

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

#### **User Interface Design or User Interface Engineering**

This is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centred design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design may be utilised to apply a theme or style to the interface without compromising its usability. The design process must balance the technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialise in certain types of projects and have skills centered around their expertise, e.g., software design, user research, web design, or industrial design.

### 3.2 Significance of User Interface

To work with a system, users have to be able to control the system and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windscreen and the exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

The term *user interface* is often used in the context of computer systems and electronic devices. The user interface of a mechanical system, a vehicle or an industrial installation is sometimes referred to as the **Human-Machine Interface (HMI)**. HMI is a modification of the original term MMI (Man-Machine Interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, which is more commonly used for Human-Computer *Interaction* than Human-Computer *Interface*. Other terms used are Operator Interface Console (OIC) and Operator Interface Terminal (OIT).

### 3.3 Types of User Interfaces

Currently (as at 2009) the following types of user interface are the most common:

- **Graphical User Interfaces (GUI)** accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-Oriented User Interfaces (OOUIs) and application oriented interfaces. Examples are Windows Operating System, LabView, etc.
- **Web-based User Interfaces or Web User Interfaces (WUI)** accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilise Java, AJAX, Adobe Flex, Microsoft, NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called control panels.



User interfaces that are common in various fields outside desktop computing:

- **Command Line Interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Examples are MS-DOS and UNIX interface. This interface is also used for system administration tasks, etc.
- **Tactile Interfaces** supplement or replace other forms of output with haptic feedback methods. This is used in automated simulators and so on.
- **Touch User Interface** are graphical user interfaces using a touchscreen display as a combined input and output device. It is used in many types of point of sale, industrial processes and machines, self-service machines etc. Examples are touch screen monitors.

Other types of user interfaces are:

- **Attentive User Interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch Interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based Interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture Interface** are graphical user interfaces which accept input in form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent User Interfaces** are human-machine interfaces that aim at improving the efficiency, effectiveness, and naturalness of

human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).

- **Motion Tracking Interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple.
- **Multi-screen Interfaces** employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Non-command User Interfaces** allow users to infer their needs and intentions, without requiring explicit commands formulation.
- **Object-Oriented User Interface (OOUI)** examples of OOUI are:

**Reflexive User Interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.

**Tangible User Interfaces** which place a greater emphasis on touch and physical environment or its element.

**Text User Interfaces** are user interfaces which output text, but accept other forms of input in addition to or in place of typed command strings.

**Voice User Interfaces** which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.

**Natural-Language Interfaces** used for search engines and on webpages. User types in a question and waits for a response.

**Zero-Input Interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.

**Zooming User Interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail

## SELF-ASSESSMENT EXERCISE

Check the GUI features and functions of LabView. Then compare their environments with MS-DOS.

### 3.4 History of User Interfaces

The history of user interfaces can be divided into the following phases according to the dominant type of user interface:

- Batch Interface, 1945-1968
- Command-line User Interface, 1969 to present
- Graphical User Interface, 1981 to present

### 3.5 User Interface Modalities and Modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

Input — allowing the users to manipulate a system. For example the computer keyboard allows the user to enter typed text; digitising tablet allows the user to create free-form drawing.

Output — allowing the system to indicate the effects of the users' manipulation. For example the computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*).

The user interface may employ several redundant input and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending on the state of the computer program. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

## 4.0 CONCLUSION

In this unit, you have been introduced to the fundamental concepts of user interface. You have also learnt the history and significance of user interface design.

## 5.0 SUMMARY

In this unit, you have learnt the:

- introduction of user interface which is the aggregate of means by which users interact with a particular machine, device, computer program or any other complex tool
- study of the various types of user interface design which includes graphical user interfaces, web-based user interfaces, command line interfaces e.t.c.
- history of user interfaces which can be divided into batch interface, command-line user interface and graphical user interface
- modality of a user interface which is a path of communication employed by the user interface to carry input and output.

## 6.0 TUTOR-MARKED ASSIGNMENT

- i. Explain the Microsoft's Clippy the paperclip.
- ii. Write a short note on the Command-line User Interface.

## 7.0 REFERENCES/FURTHER READING

Pinel, J. P. (2008). *Biopsychology* (7th ed.). Boston: Pearson. p. 357.  
Wikipedia.org

## UNIT 2 DESIGNING GOOD USER INTERFACES

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Essentials of Interface Design
    - 3.1.1 Designing a Good Interface
    - 3.1.2 Tips for Designing Good User Interface
  - 3.2 Understanding Users
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the essentials of designing good interface designs and also discusses the various users.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the essentials of a good interface design
- identify the necessary tips needed for designing a good interface
- discuss various users.

### 3.0 MAIN CONTENT

#### 3.1 Essentials of Interface Design

There are three pillars to an application's success:

- Features
- Function
- Face

*Features* refer to what the application will do for the user. Features are the requirements for the software.

*Function* refers to how well the software operates. Bug-free software will function perfectly.

**Face** refers to how the application presents itself to the user; the application's "user interface."

Features, function and face can be restated as questions:

- i. Does the software meet the user's requirements? (Features)
- ii. Does the software operate as intended? (Function)
- iii. Is the software easy to use? (Face)

The goal of user interface design is to put a happy face on the application. Stated in more concrete terms, a successful user interface will require *Zero Training* and will be *friendly not foreboding*.

### ***Zero Training***

The goal of Zero Training could be considered as a sub-goal of friendly not foreboding. However, training costs are a major impediment to the usage of software making Zero Training an important goal by itself.

There are two types of training involved in software design: software training and job training. Software training assumes the user knows how to do the job at hand and only needs to learn how to use the software to do the job. Job training teaches the user how to do the job - which can include more than how to use the application to do the job. The goal of Zero Training relates to *zero software training*. Job training can be integrated with software training, but results in a much more ambitious project.

### ***Friendly not Foreboding***

Almost everything you do to implement the goal of Zero Training will further the goal of being friendly not foreboding. However, some techniques for reducing training may slow up experienced users. For example, you could pop-up new user messages whenever the user lands in a particular field. Seeing the same message after awhile makes the experienced user dispense with the messages.

Being friendly is an attitude and encompasses more than what is necessary for the Zero Training goal. Applications do have an attitude. For example, consider the following sets of application messages:

"Database does not exist" "I could not find database "CorpInfo".  
If you are sure this name is correct,  
CorpInfo could be unavailable due to  
maintenance or LAN problems. You

should contact the help desk to see when CorpInfo will again be available.”

“SQL access error 123” “I could not save information to the database. You can try to save again to see if the error clears. If you call the help desk concerning this problem, tell them you have a “SQL access error 123”.

“out of hunk”<sup>1</sup> “I have run out of memory (RAM). This typically happens when there is a bug in the program which causes it to lose memory over time. Save your game if possible. To free the memory you will need to reset the computer (turn it off and then on).”

The attitude of the first message is “you did something that caused me a problem” while the attitude of the second message is “I have a problem. Could you give me a hand?”

### 3.1.1 Designing a Good User Interface

Designing a good user interface is an iterative process. First, you design and implement a user interface using appropriate techniques. Then you evaluate the design. The results of the evaluation feed the next design and implementation. You stop the process when you have met your design goals or you run out of time and/or money.

Note that if you have different user communities (or the same user with different jobs), you may need different user interfaces, customisable user interfaces or both. For example, Microsoft Word provides four user interfaces: normal, outline, page layout and master. In addition, Microsoft Word provides a host of customisation features for the keyboard, menu and toolbars.

While design is important, the real key to creating a good user interface is in your evaluation techniques. Obviously, you should use your own user interface. If you can't use it, how can anyone else? Next, get feedback from your testers.

The best evaluations are done by watching over the shoulder of the user. The key here is watching. If you are telling the user what to do, you will never find out if your interface is easy to use. Let the user figure it out himself or herself. If the user has to ask you what to do or pauses to figure out what to do next, you may need to work on your interface. If

the user grimaces, find out why. Learn from the experience. Some of the most innovative designs were shot down when the users could not figure them out.

You will need both new and experienced users for testing your interface. The new users will help you determine if you meet the Zero Training goal. The experienced users will let you know if your methods for meeting the Zero Training goal interfere with getting work done once the user has learned the software.

### 3.1.2 Tips for Designing a Good User Interface

- **Consistency:** The most important thing that you can possibly do is make sure that your user interface works consistently. If you can double-click on items in one list and have something happen, then you should be able to double-click on items in any other list and have the same sort of thing happen. Put your buttons in consistent places on all of your windows, use the same wording in labels and messages, and use a consistent colour scheme throughout. Consistency in your user interface allows your users to build an accurate mental model of the way that it works, and accurate mental models lead to lower training and support costs.
- **Set standards and stick to them:** The only way that you'll be able to ensure consistency within your application is to set design standards and then stick to them. The best approach is to adopt an industry standard and then fill any missing guidelines that are specific to your needs. Industry standards, such as the ones set by IBM (1993) and Microsoft (1995), will often define 95-99% of what you need. By adopting industry standards, you are not only taking advantage of the work of others, you are also increasing the chances that your application will look and feel like other applications that your users purchase or have built. User interface design standards should be set during the define infrastructure stage.
- **Explain the rules:** Your users need to know how to work with the application that you built for them. When an application works consistently, it means you only have to explain the rules once. This is a lot easier than explaining in detail exactly how to use each and every feature in an application step by step.
- **Support both novices and experts:** Although a library-catalog metaphor might be appropriate for casual users of a library system, library patrons, it probably is not all that effective for expert users, librarians. Librarians are highly trained people who



are able to use complex search systems to find information in a library; therefore you should consider building a set of search screens to support their unique needs.

- **Navigation between screens is important:** If it is difficult to get from one screen to another, then your users will quickly become frustrated and give up. When the flow between screens matches the flow of the work that the user is trying to accomplish, then your application will make sense to your users. Because different users work in different ways, your system will need to be flexible enough to support their various approaches. Interface-flow diagrams can be used during the model stage to model the flow between screens.
- **Navigation within a screen is important:** In Western societies, people read left to right and top to bottom. Because people are used to this, so you should design screens that are also organised left to right and top to bottom. All you want is to organise navigation between widgets on your screen in a manner that users will find familiar to them.
- **Word your messages and labels appropriately:** The text that you display on your screens is a primary source of information for your users. If your text is worded poorly then your interface will be perceived poorly by your users. Using full words and sentences, as opposed to abbreviations and codes makes your text easier to understand. Your messages should be worded positively, imply that the user is in control, and provide insight into how to use the application properly. For example, which message do you find more appealing “You have input the wrong information” or “An account number should be 8 digits in length?” Furthermore, your messages should be worded consistently and displayed in a consistent place on the screen. Although the messages “The person’s first name must be input.” and “An account number should be input.” are separately worded well, together they are inconsistent. In light of the first message, a better wording of the second message would be “The account number must be input” to make the two messages consistent.
- **Understand your widgets:** You should use the right widget (widgets are interface elements that the users interact with) for the right task, helping to increase the consistency in your application and probably making it easier to build the application in the first place. The only way that you can learn how to use widgets properly is to read and understand the user-interface standards and guidelines that your organisation has adopted.

- **Look at other applications with a grain of salt:** Unless you know that another application follows the user-interface standards and guidelines of your organisation, you must not assume that the application is doing things right. Although it is always a good idea to look at the work of others to get ideas, until you know how to distinguish between good and bad user-interface design you have to be careful. Too many developers make the mistake of imitating the user interface of another application that was poorly designed.
- **Use colour appropriately:** Colour should be used sparingly in your applications, and if you do use it you must also use a secondary indicator. The problem is that some of your users may be colour blind – if you are using colour to highlight something on a screen, then you need to do something else to make it stand out if you want people to notice it, such as display a symbol beside it. You also want to use colours in your application consistently so that you have a common look and feel throughout your application. Also, colour generally does not port well between platforms – what looks good on one system often looks poor on another system. We have all been to presentations where the presenter said “it looks good on my machine at home.”
- **Follow the contrast rule:** If you are going to use colour in your application, you need to ensure that your screens are still readable. The best way to do this is to follow the contrast rule: Use dark text on light backgrounds and light text on dark backgrounds. It is very easy to read blue text on a white background but very difficult to read blue text on a red background. The problem is that there is not enough contrast between blue and red to make it easy to read, whereas there is a lot of contrast between blue and white.
- **Use fonts appropriately:** Old English fonts might look good on the covers of William Shakespeare’s plays, but they are really difficult to read on a screen. Use fonts that are easy to read, such as serif fonts, Times Roman. Furthermore, use your fonts consistently and sparingly. A screen using two or three fonts effectively looks a lot better than a screen that uses five or six. Never forget that you are using a different font every time you change the size, style (bold, italics, underlining, etc.), typeface, or colour.
- **Grey things out, do not remove them:** You often find that at certain times it is not applicable to give your users access to all the functionality of an application. You need to select an object

before you can delete it, so to reinforce your mental model the application should do something with the Delete button and/or menu item. Should the button be removed or greyed out? Grey it out, never remove it. By greying things out when they shouldn't be, users can start building an accurate mental model as to how your application works. If you simply remove a widget or menu item instead of greying it out then it is much more difficult for your users to build an accurate mental model, because they only know what is currently available and not what is not available to them. The old adage that out of sight is out of mind is directly applicable here.

- **Use non destructive default buttons:** It is quite common to define a default button on every screen, the button that gets invoked if the user presses the return/enter key. The problem is that sometimes people will accidentally hit the enter/return key when they do not mean to, consequently invoking the default button. Your default button shouldn't be something that is potentially destructive, such as delete or save (perhaps your user really did not want to save the object at that moment).
- **Alignment of fields:** When a screen has more than one editing field, you need to organise the fields in a way that is both visually appealing and efficient. As shown in figure 1.I, the best way to do so is to left-justify edit fields, or in other words make the left-hand side of each edit field line up in a straight line, one over the other. The corresponding labels should be right justified and placed immediately beside the field. This is a clean and efficient way to organise the fields on a screen.
- **Justify data appropriately:** For columns of data it is common practice to right justify integers, decimal align floating point numbers, and left justify strings.
- **Do not create busy screens:** Crowded screens are difficult to understand and hence are difficult to use. Experimental results show that the overall density of the screen should not exceed 40%, whereas local density within groupings shouldn't exceed 62%.
- **Group things on the screen effectively:** Items that are logically connected should be grouped together on the screen to communicate that they are connected, whereas items that have nothing to do with each other should be separated. You can use whitespace between collections of items to group them and/or you can put boxes around them to accomplish the same thing.

- **Open windows in the centre of the action:** When your user double-clicks on an object to display its edit/detail screen then his or her attention is on that spot. Therefore it makes sense to open the window in that spot, not somewhere else.
- **Pop-up menus should not be the only source of functionality:** Your users cannot learn how to use your application if you hide major functionality from them. One of the most frustrating practices of developers is to misuse pop-up, also called context-sensitive, menus. Typically there is a way to use the mouse on your computer to display a hidden pop-up menu that provides access to functionality that is specific to the area of the screen that you are currently working on.

**Poor Alignment**

Name:

Date of Birth:

ID.....:

**Good Alignment**

Name:

Date of Birth:

ID:

**Fig. 1.1: Showing that alignment of fields is critical**

### 3.2 Understanding Users

You must understand the user to be able to put a happy face on your application. You should understand the user's job, how the software fits in with that job and how the user goes about getting the job done. You need to approach the design of software from the user's viewpoint not from an abstract requirements document. Specifically, you should understand what the user will be doing with the application. If you can think like a user, you can create a much better user interface.

Here are some basic principles to remember about users:

- Your software is like a hammer - The user doesn't really care how well crafted it is, the user just wants nails put in the wall. Users just want to do their job (or play their game). They don't

care about you or your software. Your software is just an expedient tool to take the user where the user wants to go.

- Given a selection of hammers to buy at the hardware store - The user will select the one which will be most fun to use. Of course, this varies by user - some will want the plastic handle, some the wood, some the green, etc. When evaluating your software, users are often swayed by looks, not function. Thus, steps taken to make the product look good (nice icons, pictures, good colour scheme, fields aligned, etc.) will often favourably enhance evaluations of your software.
- It had better drive nails - The user will not really know if your software is adequate to the job until the user has used the software to do the actual work. From an interface perspective, the software should not look like it can do more than it can.
- Some users will try to use a hammer to drive a screw - If your software is good, some user somewhere will try to use the software for some purpose for which you never intended it to be used. Obviously, you cannot design a user interface to deal with uses you cannot foresee. There is no single rigid model of the right way to use the software, so build in flexibility.
- Users will not read an instruction manual for a hammer- They won't read one for your software either, unless they really have to. Users find reading instruction manuals almost as unpleasurable as dental work.
- A user reading the instruction manual for a hammer is in serious trouble - When you create your help system (and printed manual), remember that the user will only resort to those materials if he or she is in trouble. The user will want a problem solved as fast and as easily as possible.
- Hammers don't complain - You should try to eliminate error messages and any error messages your program needs should have the right attitude.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the essentials of good interface design. You have also learnt the necessary tips needed for designing a good interface and the need for understanding various users.

## 5.0 SUMMARY

In this unit, we have learnt:

- the essentials of interface design with emphasis on the features, functions and the face of the software
- that designing a good user interface which has been described as an iterative process involves designing, implementing, evaluating and redesigning until all removable errors have been taken care of
- the tips necessary for designing a good user interface which includes consistency, setting standards and sticking to them, supporting of both novices and experts, e.t.c.
- understanding the user's job, how the software fits in with that job and how the user goes about getting the job done.

### SELF- ASSESSMENT EXERCISE

- i. How do you ensure that the interface supports both novices and experts?
- ii. Write short notes on any three tips necessary for designing a good user interface.

## 6.0 TUTOR- MARKED ASSIGNMENT

- i. How do you ensure that the interface support both novices and expert?
- ii. Write short note on the design of a user interface for a user with hearing disability.

## 7.0 REFERENCES/FURTHER READING

Wikipedia.org

[www://www.linfo.org/gui.html](http://www.linfo.org/gui.html)

## UNIT 3 GRAPHICAL USER INTERFACE

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Graphical User Interface
  - 3.2 History of Graphical User Interface
  - 3.3 Elements of Graphical User Interface
  - 3.4 Three Dimensional Graphical User Interface
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

This unit describes the general concept of Graphical User Interface (GUI) and also the history and elements of graphical user interface. The concept of three dimensional (3D) graphical user interface is also introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe a graphical user interface
- explain the history behind graphical user interface
- list the elements of a graphical user interface
- describe the three-dimensional user interfaces.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Graphical User Interface

Graphical User Interfaces, also known as GUIs, offer a consistent visual language to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. This explains the most common elements of the visual language interfaces.

A **graphical user interface** is a type of user interface which allows people to interact with electronic devices such as computers; hand-held devices such as MP3 Players, Portable Media Players or Gaming

devices; household appliances and office equipment with images rather than text commands. A *GUI* offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

The term *GUI* is historically restricted to the scope of two-dimensional display screens with display resolutions capable of describing generic information, in the tradition of the computer science research at Palo Alto Research Centre (PARC). The term *GUI* earlier might have been applicable to other high-resolution types of interfaces that are non-generic, such as videogames, or not restricted to flat screens, like volumetric displays.

## 3.2 History of Graphical User Interface

### Precursor to GUI

The precursor to GUIs was invented by researchers at the Stanford Research Institute, led by Douglas Engelbart. They developed the use of text-based hyperlinks manipulated with a mouse for the On-Line System. The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC, who went beyond text-based hyperlinks and used a GUI as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. As a result, some people call this class of interface PARC User Interface (PUI) (note that PUI is also an acronym for Perceptual User Interface). Ivan Sutherland developed a pointer-based system called the Sketchpad in 1963. It used a light-pen to guide the creation and manipulation of objects in engineering drawings.

### PARC User Interface

The PARC User Interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC User Interface employs a pointing device in addition to a keyboard. These aspects can be emphasised by using the alternative acronym WIMP, which stands for **W**indows, **I**cons, **M**enus and **P**ointing device.

### Evolution

Following PARC, the first GUI-centric computer operating model was the Xerox 8010 Star Information System in 1981 followed by the Apple Lisa (which presented concept of menu bar as well as window controls) in 1982 and the Atari ST and Commodore Amiga in 1985.



The GUIs that are familiar to most people today are Microsoft Windows, Finder Interface (Mac OS X), and the X Window System interfaces. Apple, IBM and Microsoft used many of Xerox's ideas to develop products, and IBM's Common User Access specifications formed the basis of the user interface found in Microsoft Windows, IBM OS/2 Presentation Manager, and the Unix Motif toolkit and window manager. These ideas evolved to create the interface found in current versions of Microsoft Windows, as well as in Mac OS X and various desktop environments for Unix-like operating systems, such as Linux. Thus most current GUIs have largely common idioms.

### **Post-WIMP Interfaces**

Smaller mobile devices such as PDAs and smart phones typically use the WIMP elements with different unifying metaphors, due to constraints in space and available input devices. Applications for which WIMP is not well suited may use newer interaction techniques, collectively named as post-WIMP user interfaces.

Some touch-screen-based operating systems such as Apple's iPhone OS currently use post-WIMP styles of interaction. The iPhone's use of more than one finger in contact with the screen allows actions such as pinching and rotating, which are not supported by a single pointer and mouse.

A class of GUIs sometimes referred to as post-WIMP include 3D compositing window manager such as Compiz, Desktop Window Manager, and LG3D. Some post-WIMP interfaces may be better suited for applications which model immersive 3D environments, such as Google Earth.

## **3.3 Elements of Graphical User Interfaces**

A GUI uses a combination of technologies and devices to provide a platform which the user can interact with in order to achieve the tasks of gathering and producing information.

Series of elements conforming to visual languages have evolved to represent information stored in computers. This makes it easier for people with little computer skills to work with and use computer software. The most common combination of such elements in GUIs is the WIMP paradigm, especially in personal computers.

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of

the static elements on which the user can interact, and define the appearance of the interface.

The key elements of GUI are divided into two categories viz Structural and Interactive elements.

### 3.3.1 Structural Elements

User interfaces use visual conventions to represent the generic information shown. Some conventions are used to build the structure of the static elements on which the user can interact, and define the appearance of the interface.

#### Window

A window is an area on the screen that displays information, with its contents being displayed independently from the rest of the screen. An example of a window is what appears on the screen when the "My Documents" icon is clicked in the Windows Operating System. It is easy for a user to manipulate a window: it can be opened and closed by clicking on an icon or application, and it can be moved to any area by dragging it (that is, by clicking in a certain area of the window – usually the title bar along the top – and keeping the pointing device's button pressed, then moving the pointing device). A window can be placed in front or behind another window, its size can be adjusted, and scrollbars can be used to navigate the sections within it. Multiple windows can also be opened at one time, in which case each window can display a different application or file – this is very useful when working in a multitasking environment. The system's memory is the only limitation to the number of windows that can be opened at once. There are also many types of specialised windows.

- A **Container Window**: a window that is opened while invoking the icon of a mass storage device, directory or folder and which is presenting an ordered list of other icons that could be again some other directories, or data files or may be even executable programs. All modern container windows could present their content on screen either by acting as browser windows or text windows. Their behaviour can automatically change according to the choices of the single users and their preferred approach to the graphical user interface.
- A **browser window**: allows the user to move forward and backward through a sequence of documents or web pages. Web browsers are examples of these types of windows.
- **Text terminal windows**: are designed for embedding interaction with text user interfaces within the overall graphical interface.

MS-DOS and UNIX consoles are examples of these types of windows.

- A **child window**: opens automatically or as a result of a user activity in a parent window (A parent window can be any type of window). Pop-up windows on the Internet can be child windows.
- A **message window**, or **dialog box**: is a type of child window. These are usually small and basic windows that are opened by a program to display information to the user and/or get information from the user. They usually have a button that must be pushed before the program can be resumed.

## Menus

Menus allow the user to execute commands by selecting from a list of choices. Options are selected with a mouse or other pointing device within a GUI. A keyboard may also be used. Menus are convenient because they show what commands are available within the software. This limits the amount of documentation the user reads to understand the software.

- A **menu bar** is displayed horizontally across the top of the screen and/or along the top of some or all windows. A pull-down menu is commonly associated with this menu type. When a user clicks on a menu option, the pull-down menu will appear.
- A **menu** has a visible title within the menu bar. Its contents are only revealed when the user selects it with a pointer. The user is then able to select the items within the pull-down menu. When the user clicks elsewhere, the contents of the menu will disappear.
- A **context menu** is invisible until the user performs a specific mouse action, like pressing the right mouse button. When the software-specific mouse action occurs the menu will appear under the cursor.
- **Menu extras** are individual items within or at the side of a menu.

## Icons

An icon is a small picture that represents objects such as a file, program, web page, or command. Icons are used as a quick way to execute commands, open documents, and run programs. They are also very useful when searching for an object in a browser list, because in many operating systems all documents using the same extension will have the same icon.

## **Controls (or Widgets)**

The interface element that a computer user interacts with is known as a **control** or **widget**.

### *Window*

A paper-like rectangle that represents a "window" into a document, form, or design area.

### *Pointer (or mouse cursor)*

The spot where the mouse "cursor" is currently referencing.

### *Text box*

A box in which texts or numbers are entered.

### *Button*

An equivalent to a push-button as found on mechanical or electronic instruments.

### *Hyperlink*

Text with some kind of indicators (usually underlining and/or colour) that shows that clicking it will take one to another screen or page.

### *Drop-down list*

A list of items from which to select: The list normally only displays items when a special button or indicator is clicked.

### *Check box*

A box which indicates an "on" or "off" state via a check-mark or an "×".

### *Radio button*

A button, similar to a check-box, except that only one item in a group can be selected. Its name comes from the mechanical push-button group on a car radio receiver. Selecting a new item from the group's buttons also deselects the previously selected button.

### *Data grid*

A spreadsheet-like grid that allows numbers or text to be entered in rows and columns.

## **Tabs**

A tab is typically a rectangular small box which usually contains a text label or graphical icon associated with a view pane. When activated the view pane, or window, displays widgets associated with that tab; groups of tabs allow the user to switch quickly between different widgets. This

is used in the web browsers Firefox, Internet Explorer, Konqueror, Opera, and Safari. With these browsers, you can have multiple web pages open at once in one window, and quickly navigate between them by clicking on the tabs associated with the pages. Tabs are usually placed in groups at the top of a window, but may also be grouped on the side or bottom of a window.

### **3.3.2 Interaction Elements**

Some common idioms for interaction have evolved in the visual language used in GUIs. Interaction elements are interface objects that represent the state of an ongoing operation or transformation, either as visual reminders of the user intent (such as the pointer), or as affordances showing places where the user may interact.

#### **Cursor**

A cursor is an indicator used to show the position on a computer monitor or other display devices that will respond to inputs from a text input or pointing devices.

#### **Pointer**

One of the most common components of a GUI on the personal computer is a pointer: a graphical image on a screen that indicates the location of a pointing device, and can be used to select and move objects or commands on the screen. A pointer commonly appears as an angled arrow, but it can vary within different programs or operating systems. Example of this can be found within text-processing applications, which use an I-beam pointer that is shaped like a capital I, or in web browsers, which often indicate that the pointer is over a hyperlink by turning the pointer in the shape of a gloved hand with outstretched index finger.

The use of a pointer is employed when the input method, or pointing device, is a device that can move fluidly across a screen and select or highlight objects on the screen. Pointer trails can be used to enhance its visibility during movement. In GUIs where the input method relies on hard keys, such as the five-way key on many mobile phones, there is no pointer employed, and instead the GUI relies on a clear focus state.

#### **Selection**

A selection is a list of items on which user operations will take place. The user typically adds items to the list manually, although the computer may create a selection automatically.

## Adjustment handle

A handle is an indicator of a starting point for a drag and drop operation. Usually the pointer shape changes when placed on the handle, showing an icon that represents the supported drag operation.

## SELF-ASSESSMENT EXERCISE 1

Identify and study these elements within Window operating system

### 3.4 Three-dimensional User Interfaces

For typical computer displays, *three-dimensional* are a misnomer—their displays are two-dimensional. Three-dimensional images are projected on them in two dimensions. Since this technique has been in use for many years, the recent use of the term three-dimensional must be considered a declaration by equipment marketers that the speed of three dimensions to two dimensions projection is adequate to use in standard GUIs.

#### Motivation

Three-dimensional GUIs are quite common in science fiction literature and movies, such as in *Jurassic Park*, which features Silicon Graphics' three-dimensional file manager, "File system navigator", an actual file manager that never got much widespread use as the user interface for a Unix computer. In fiction, three-dimensional user interfaces are often immersible environments like William Gibson's *Cyberspace* or Neal Stephenson's *Metaverse*.

Three-dimensional graphics are currently mostly used in computer games, art and computer-aided design (CAD). There have been several attempts at making three-dimensional desktop environments like Sun's Project Looking Glass or SphereXP from Sphere Inc. A three-dimensional computing environment could possibly be used for collaborative work. For example, scientists could study three-dimensional models of molecules in a virtual reality environment, or engineers could work on assembling a three-dimensional model of an airplane. This is a goal of the Croquet project and Project Looking Glass.

#### Technologies

The use of three-dimensional graphics has become increasingly common in mainstream operating systems, from creating attractive interfaces—eye candy—to functional purposes only possible using three dimensions. For example, user switching is represented by rotating

a cube whose faces are each user's workspace, and window management is represented in the form or via a Rolodex-style flipping mechanism in Windows Vista (see Windows Flip 3D). In both cases, the operating system transforms windows on-the-fly while continuing to update the content of those windows.

Interfaces for the X Window System have also implemented advanced three-dimensional user interfaces through compositing window managers such as Beryl, Compiz and KWin using the AIGLX or XGL architectures, allowing for the usage of OpenGL to animate the user's interactions with the desktop.

Another branch in the three-dimensional desktop environment is the three-dimensional GUIs that take the desktop metaphor a step further, like the BumpTop, where a user can manipulate documents and windows as if they were "real world" documents, with realistic movement and physics.

The Zooming User Interface (ZUI) is a related technology that promises to deliver the representation benefits of 3D environments without their usability drawbacks of orientation problems and hidden objects. It is a logical advancement on the GUI, blending some three-dimensional movement with two-dimensional or "2.5D" vector objects.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to graphical user interface (GUI). The history of graphical user interface was also discussed. The elements of graphical user interface were also explained. You were also introduced to three-dimensional user interfaces.

#### **5.0 SUMMARY**

In this unit, you have learnt the:

- introduction to graphical user interface which is a type of user interface that allows people to interact with electronic devices such as computers, hand-held devices, household appliances and office equipment with images rather than text commands
- history of graphical user interface, precursor to GUI, PARC user interface and the evolution of other graphical user interfaces
- elements of graphical user interface which are divided into two categories that includes structural and interactive elements.

## **SELF-ASSESSMENT EXERCISE 2**

- i. What do you understand by graphical user interface?
- ii. Explain the structural and iterative elements of graphical user interface.

## **6.0 TUTOR- MARKED ASSIGNMENT**

Explain the PARC graphical user interface

## **7.0 REFERENCES/FURTHER READING**

<http://www.linfo.org/gui.html>. Retrieved on 12 November 2008.

[www.wikipedia.com](http://www.wikipedia.com)



## UNIT 4 HUMAN- COMPUTER INTERACTION

### CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Human -Computer Interaction (HCI)
  - 3.2 Goals of HCI
  - 3.3 Differences with Other Related Fields
  - 3.4 Future Development of HCI
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor -Marked Assignment
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In this unit, you will be introduced to Human-Computer Interaction (HCI) and its differences with other related fields. The goals and future development of human- computer interaction and the general concept of human-computer interface will be introduced.

### 2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the term human- computer interaction
- identify the various goals of human -computer interaction
- differentiate human -computer interaction from other related fields
- describe the future development of HCI and explain the human - computer interface.

### 3.0 MAIN CONTENT

#### 3.1 Introduction to Human -Computer Interaction

**Human–Computer Interaction (HCI)** is the study of interaction between people (users) and computers. It involves the study intersection of computer science, behavioural sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply *interface*), which includes both software and hardware. The association for computing machinery defines *human-computer interaction* as "a discipline concerned with the design,

evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them” Since human-computer interaction studies a human and a machine together, it draws its supporting knowledge from the machine and human sides. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, statistics, linguistics, social sciences, cognitive psychology, and human performance are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success. HCI is sometimes referred to as **Man–Machine Interaction (MMI)** or **Computer–Human Interaction (CHI)**.

### 3.1.1 Human–Computer Interface

The human–computer interface can be described as the point of communication between the human user and the computer. The flow of information between the human and computer is defined as the loop of interaction. The loop of interaction has several aspects to it including:

- **Task Environment:** The conditions and goals set upon the user.
- **Machine Environment:** The environment that the computer is connected to, i.e. a laptop in a college student's dorm room.
- **Areas of the Interface:** Non-overlapping areas involve processes of the human and computer not pertaining to their interaction. Meanwhile, the overlapping areas only concern themselves with the processes pertaining to their interaction.
- **Input Flow:** Begins in the task environment as the user has some task that requires using their computer.
- **Output:** The flow of information that originates in the machine environment.
- **Feedback:** Loops through the interface that evaluate, moderate, and confirm processes as they pass from the human through the interface to the computer and back.

### 3.2 Goals of HCI

The basic goal of HCI is to improve the interactions between users and computers by making computers more usable and responsive to the user's needs. Specifically, HCI is concerned with:

- methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface

within given constraints, optimising for a desired property such as learning ability or efficiency of use)

- methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- techniques for evaluating and comparing interfaces
- developing new interfaces and interaction techniques
- developing descriptive and predictive models and theories of interaction.

A long term goal of HCI is to design systems that minimise the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

Researchers in HCI are interested in developing new design methodologies, experimenting with new hardware devices, prototyping new software systems, exploring new paradigms for interaction, and developing models and theories of interaction.

### **3.3 Differences with Related Fields**

HCI focuses on user interface design mainly for users of computer system and promotes effective interaction between computers and users (human). User Interface Design is concerned with the users of devices such as computers, appliances, machines, mobile communication devices, software applications, and websites. In HCI, efficient user interface is critical.

HCI differs from human factors in that the focus is more on users working specifically with computers, rather than other kinds of machines or designed artifacts. There is also a focus in HCI on how to implement the computer software and hardware mechanisms to support human-computer interaction. Thus, human factors are a broader term; HCI could be described as the human factors of computers, although some experts try to differentiate these areas.

According to some experts, HCI also differs from ergonomics (which will be fully introduced in the next unit) in that there is less focus on repetitive work-oriented tasks and procedures, and much less emphasis on physical stress and the physical form or industrial design of the user interface, such as keyboards and mice. However, this does not take full account of ergonomics, which recently has gained a much broader focus (equivalent to human factors). Cognitive ergonomics, for example, is a

part of ergonomics, of which *software ergonomics* (an older term, essentially the same as HCI) is a part.

Three areas of study have substantial overlap with HCI even as the focus of inquiry shifts. In the study of Personal Information Management (PIM), human interactions with the computer are placed in a larger informational context - people may work with many forms of information, some computer-based, many not (e.g., whiteboards, notebooks, sticky notes, refrigerator magnets) in order to understand and effect desired changes in their world. In Computer Supported Cooperative Work (CSCW), emphasis is placed on the use of computing systems in support of the collaborative work of a group of people. The principles of Human Interaction Management (HIM) extend the scope of CSCW to an organisational level which can be implemented without the use of computer systems.

### 3.4 Future Development of HCI

As the means by which humans interact with computers continues to evolve rapidly, human-computer interaction is affected by the forces shaping the nature of future computing. These forces include:

- decreasing hardware costs leading to larger memories and faster systems
- miniaturisation of hardware leading to portability
- reduction in power requirements leading to portability
- new display technologies leading to the packaging of computational devices in new forms
- specialised hardware leading to new functions
- increased development of network communication and distributed computing
- increasingly widespread use of computers, especially by people who are outside the computing profession
- increasing innovation in input techniques (i.e., voice, gesture, pen), combined with lowering cost, leading to rapid computerisation by people previously left out of the "computer revolution"
- wider social concerns leading to improved access to computers by currently disadvantaged groups.

The future for HCI is expected to include the following features:

**Ubiquitous communication:** Computers will communicate through high speed local networks, nationally over wide-area networks, and portably via infrared, ultrasonic, cellular, and other technologies. Data

and computational services will be portably accessible from many if not most locations to which a user travels.

**High functionality systems:** Systems will have large numbers of functions associated with them. There will be so many systems that most users, technical or non-technical, will not have time to learn them in the traditional way (e.g., through thick manuals).

**Mass availability of computer graphics:** Computer graphics capabilities such as image processing, graphics transformations, rendering, and interactive animation will become widespread as inexpensive chips become available for inclusion in general workstations.

**Mixed media:** Systems that will handle images, voice, sounds, video, text, formatted data. These will be exchangeable over communication links among users. The separate worlds of consumer electronics (e.g., stereo sets, VCRs, televisions) and computers will partially merge. Computer and print worlds will continue to cross assimilate each other.

**High-bandwidth interaction:** The rate at which humans and machines interact will increase substantially due to the changes in speed, computer graphics, new media, and new input/output devices. This will lead to some qualitatively different interfaces, such as virtual reality or computational video.

**Large and thin displays:** New display technologies will finally mature enabling very large displays and also displays that are thin, light weight, and have low power consumption. This will have large effects on portability and will enable the development of paper-like, pen-based computer interaction systems very different in feel from desktop workstations of the present.

**Embedded computation:** Computation will pass beyond desktop computers into every object for which uses can be found. The environment will be alive with little computations from computerised cooking appliances. to lighting and plumbing fixtures to window blinds to automobile braking systems to greeting cards. To some extent, this development is already taking place. The difference in the future is the addition of networked communications that will allow many of these embedded computations to coordinate with each other and with the user. Human interfaces to these embedded devices will in many cases be very different from those appropriate to workstations.

**Augmented reality:** A common staple of science fiction, augmented reality refers to the notion of layering relevant information into our

vision of the world. Existing projects show real-time statistics to users performing difficult tasks, such as manufacturing. Future work might include augmenting our social interactions by providing additional information about those we converse with.

**Group interfaces:** Interfaces to allow groups of people to coordinate will be common (e.g., for meetings, for engineering projects, for authoring joint documents). These will have major impacts on the nature of organisations and on the division of labour. Models of the group design process will be embedded in systems and will cause increased rationalisation of the design.

**User tailorability:** Ordinary users will routinely tailor applications to their own use and will use this power to invent new applications based on their understanding of their own domains. Users, with their deeper knowledge of their own knowledge domains, will increasingly be important sources of new applications at the expense of generic systems programmers (with systems expertise but low domain expertise).

**Information utilities:** Public information utilities (such as home banking and shopping) and specialised industry services (e.g., weather for pilots) will continue to proliferate. The rate of proliferation will accelerate with the introduction of high-bandwidth interaction and the improvement in quality of interfaces.

## 4.0 CONCLUSION

In this unit, you have been introduced to the concepts of human - computer interaction. You have also been introduced to the various goals of HCI and also the difference between HCI and other related fields. The future of HCI was also discussed. The concept of the human-computer interface was also introduced.

## 5.0 SUMMARY

In this unit, you have learnt the following:

- introduction to human-computer interaction (HCI) which is the study of interaction between people (users) and computers. HCI is also sometimes referred to as man-machine interaction (MMI) or computer-human interaction (CHI)
- the human-computer interface which can be described as the point of communication between the human user and the computer

- the goal of HCI which is basically to improve the interactions between users and computers by making computers more usable and receptive to the user's needs
- the difference between HCI and other related fields like graphical user interface, ergonomics e.t.c.
- the future development in HCI like Ubiquitous communication, High functionality systems, Mass availability of computer graphics e.t.c.

### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by Human -Computer Interaction?
- ii. Describe the Human -Computer Interface

### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly the future development in HCI.

### **7.0 REFERENCES/FURTHER READING**

More discussion of the differences between these terms can be found in the ACM SIGCHI Curricula for Human-Computer Interaction

Brown, C. M. (1998). *Human-Computer Interface Design Guidelines*. Intellect Books. pp.2–3.

Green, P. (2008). Iterative Design. Lecture presented in Industrial and Operations Engineering 436 Human Factors in Computer Systems, University of Michigan, Ann Arbor, MI, February 4, 2008.

Wickens, C. D., John, D. L, Yili, L., & Sallie, E. G. (2004). *An Introduction to Human Factors Engineering*. (2nd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. pp.185–193.

## **UNIT 5     ERGONOMICS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Introduction to Ergonomics
  - 3.2 Five Aspects of Ergonomics
  - 3.3 History of Ergonomics
  - 3.4 Ergonomics in Workplace
  - 3.5 Efficiency and Ergonomics
  - 3.6 Benefits of Ergonomics
  - 3.7 Fields of Ergonomics
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

This unit describes in great detail the various aspects of ergonomics alongside the history behind it. The efficiency and benefits of ergonomics will also be discussed. Different fields of ergonomics will also be highlighted.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- define the term ergonomics
- identify the various aspects of ergonomics
- explain the history of ergonomics
- describe efficiency and ergonomics
- identify the various benefits of ergonomics
- list the various fields of ergonomics.

### **3.0 MAIN CONTENT**

#### **3.1 Introduction to Ergonomics**

Ergonomics is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work.



The International Ergonomics Association has adopted this technical definition: ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimise human well-being and overall system performance.

Ergonomics is the science of making things comfy. It also makes things efficient. However for simplicity, ergonomics makes things comfortable and efficient.

At its simplest definition, ergonomics literally means the science of work. So ergonomists, i.e. the practitioners of ergonomics, study work, how work is done and how to work better. It is the attempt to make work better that ergonomics becomes so useful.

However, what you, or the user, is most concerned with is, “How can I use the product or service, will it meet my needs, and will I like using it?” Ergonomics helps define how it is used, how it meets you needs, and most importantly if you like it. It makes things comfy and efficient.

Ergonomics is concerned with the ‘fit’ between people and their work. It takes account of the worker's capabilities and limitations in seeking to ensure that tasks, equipment, information and the environment suit each worker.

To assess the fit between a person and his/her work, ergonomists consider the job being done and the demands on the worker; the equipment used (its size, shape, and how appropriate it is for the task), and the information used (how it is presented, accessed, and changed). Ergonomics draws on many disciplines in its study of humans and their environments, including anthropometry, biomechanics, mechanical engineering, industrial engineering, industrial design, kinesiology, physiology and psychology.

### **3.2 Five Aspects of Ergonomics**

There are five aspects of ergonomics: safety, comfort, ease of use, productivity/performance, and aesthetics. From these aspects of ergonomics, examples are given of how products or systems could benefit from redesign based on ergonomic principles.

- Safety – This has to do with the ability to use a device or work with a device without short or long term damage to parts of the body. For example in Medicine bottles: The print on them could be larger so that a sick person who may have bad vision (due to

sinuses, etc.) can easily see the dosages and label. Ergonomics could design the print style, colour and size for optimal viewing.

- Comfort – Comfort in the human-machine interface is usually noticed first. Physical comfort in how an item feels is pleasing to the user. If you do not like to touch it you won't. If you do not touch it you will not operate it. If you do not operate it, then it is useless. For example, in Alarm clock display: Some displays are harshly bright, drawing one's eye to the light when surroundings are dark. Ergonomic principles could re-design this based on contrast principles.
- Ease of use – This has to do with the ability to use a device with no stress. For example in Street Signs: In a strange area, many times it is difficult to spot street signs. This could be addressed with the principles of visual detection in ergonomics.
- Productivity/performance – For example in High-definition Television (HD TV): The sound on HD TV is much lower than regular TV. So when you switch from HD to regular, the volume increases dramatically. Ergonomics recognises that this difference in decibel level creates a difference in loudness and hurts human ears and this could be solved by evening out the decibel levels.
- Aesthetics - the look and feel of the object, the user experience.

### 3.3 History of Ergonomics

The foundations of the science of ergonomics appear to have been laid within the context of the culture of Ancient Greece. A good deal of evidence indicates that Hellenic civilisation in the 5th century BC used ergonomic principles in the design of their tools, jobs, and workplaces.

The term ergonomics is derived from the Greek words *ergon* [work] and *nomos* [natural laws] and first entered the modern lexicon when Wojciech Jastrzębowski used the word in his 1857 article *Rys ergonomji czyli nauki o pracy, opartej na prawdach poczerpniętych z Nauki Przyrody* (The Outline of Ergonomics, i.e. Science of Work, Based on the Truths Taken from the Natural Science).

Later, in the 19th century, Frederick Winslow Taylor pioneered the "Scientific Management" method, which proposed a way to find the optimum method for carrying out a given task. Taylor found that he could, for example, triple the amount of coal that workers were shoveling by incrementally reducing the size and weight of coal shovels

until the fastest shoveling rate was reached. Frank and Lillian Gilbreth expanded Taylor's methods in the early 1900s to develop "Time and Motion Studies". They aimed at improving efficiency by eliminating unnecessary steps and actions. By applying this approach, the Gilbreths reduced the number of motions in bricklaying from 18 to 4.5, allowing bricklayers to increase their productivity from 120 to 350 bricks per hour.

World War II marked the development of new and complex machines and weaponry, and these made new demands on operators' cognition. The decision-making, attention, situational awareness and hand-eye coordination of the machine's operator became keys to the success or failure of a task. It was observed that fully functional aircraft, flown by the best-trained pilots, still crashed. In 1943, Alphonse Chapanis, a lieutenant in the U.S. Army, showed that this so-called "pilot error" could be greatly reduced when more logical and differentiable controls replaced confusing designs in airplane cockpits. The experience of the World War II promoted the study. Man-Machine interaction became a major focus when weapons meant to attack the enemies were malfunctioning – attacking friends instead of foes.

In the decades since the war, ergonomics has continued to flourish and diversify. The Space Age created new human factors issues such as weightlessness and extreme g-forces. How far could environments in space be tolerated, and what effects would they have on the mind and body? The dawn of the Information Age has resulted in the new ergonomics field of human-computer interaction (HCI). Likewise, the growing demand for and competition among consumer goods and electronics has resulted in more companies including human factors in product design.

At home, work, or play new problems and questions must be resolved constantly. People come in all different shapes and sizes, and with different capabilities and limitations in strength, speed, judgment, and skills. All of these factors need to be considered in the design function. To solve design problems, physiology and psychology must be included with an engineering approach.

### 3.4 Ergonomics in Workplace



**Fig. 5.1: Description of Workplace Environment**

(Source: [www.wikipedia.org](http://www.wikipedia.org))

Fundamentals for the Flexible Workplace variability and compatibility with desk components, that flex from individual work activities to team settings. Workstations provide supportive ergonomics for task-intensive environments.

Outside of the discipline itself, the term 'ergonomics' is generally used to refer to physical ergonomics as it relates to the workplace (example ergonomic chairs and keyboards). Ergonomics in the workplace has to do largely with the safety of employees, both in the long and short-term. Ergonomics can help reduce costs by improving safety. This would decrease the money paid out in workers' compensation.

Workplaces may either take the reactive or proactive approach when applying ergonomics practices. Reactive ergonomics is when something needs to be fixed, and corrective action is taken. Proactive ergonomics is the process of seeking areas that could be improved and fixing the issues before they become a large problem. Problems may be fixed through equipment design, task design, or environmental design. Equipment design changes the actual, physical devices used by people. Task design changes what people do with the equipment. Environmental design changes the environment in which people work, but not the physical equipment they use.

### 3.5 Efficiency and Ergonomics

Efficiency is quite simply making something easier to do. Several forms of efficiency are:

- Reducing the strength required makes a process more physically efficient.
- Reducing the number of steps in a task makes it quicker (i.e. efficient) to complete.
- Reducing the number of parts makes repairs more efficient.
- Reducing the amount of training needed, i.e. making it more intuitive, gives you a larger number of people who are qualified to perform the task. Imagine how in-efficient trash disposal would be if your teenage child wasn't capable of taking out the garbage. Have you tried an ergonomic trash bag?

Efficiency can be found almost everywhere. If something is easier to do you are more likely to do it. If you do it more, then it is more useful. Again, utility is the only true measure of the quality of a design. And if you willingly do something more often you have a greater chance of liking it. If you like doing it you will be more comfortable doing it. So the next time you hear the term ergonomics you will know what it means to you. And I hope that is a comforting thought.

Ergonomics can help you in many ways. Among other things, it can benefit your life, health, productivity and accuracy. One of the best benefits of ergonomics is saving time. We never seem to have enough of it as it is, so why not try to get a little more out of your day? Ergonomics is about making things more efficient. By increasing the efficiency of a tool or a task, you tend to shorten the length of time it takes to accomplish your goal.

### **3.6 Benefits of Ergonomics**

The three main benefits of ergonomics are:

#### **Slim Down the Task**

Have you ever wondered why some things are so convoluted, cumbersome and chaotic? And they take forever to complete. And most of what you do does not aid the outcome.

For example, think back to the last time you got hired for a job, bought a house or car, or did something else that required a ton of paperwork. How many different forms did you write the same information on? That was not very ergonomic.

You can almost always make a task a little leaner. But first you have to understand the task. Ergonomics requires that tasks are properly slimed down and steps involved in a task are well written out. This is why task

analysis is normally done in ergonomics (Task analysis is fully discussed in module 2, unit 2).

Once you have all the steps written out, you need to take a good look at them and identify areas that you can "ergonomise":

1. *Repetition* – Look for steps that are repeated and see if they are all necessary.
2. *Order* – See if you can re-order the steps to optimise your effort.
3. *Synergy* – Can you combine things or somehow get more bang for your buck?
4. *Value Added* – Look at every step and make sure it adds value to the outcome. If it doesn't, cut it.
5. *Necessity* – Make sure the quantity of the step is needed. Do you really need to brush your teeth with 57 strokes, or will 32 do?

### **Simplify the Task**

You can also save time by simplifying the task. This is not just about reducing the number of steps, but making those steps easier to perform. The less training and/or skill that is required for a task, the quicker the pace at which it tends to get finished.

This is a great ergonomic tip, especially when the task requires more than one person. If you are trying to get your kids to pick up their toys before they go to bed, you can save a lot of time by making it easier. That is what the toy chest is for. Instead of having different places for different things, they can just throw everything in one place.

### **Increase Body Mechanism**

Ergonomic can increase your body mechanics. A good ergonomic tool acts as extension of your body enhancing capabilities. Some tools make you more effective and faster at completing a task. (Try cutting a log without an axe and see how long it takes you.)

## **3.7 Fields of Ergonomics**

### **Engineering Psychology**

*Engineering psychology* is an interdisciplinary part of Ergonomics and studies the relationships of people to machines, with the intent of improving such relationships. This may involve redesigning equipment, changing the way people use machines, or changing the location in which the work takes place. Often, the work of an engineering

psychologist is described as making the relationship more "user-friendly."

Engineering Psychology is an applied field of psychology concerned with psychological factors in the design and use of equipment. Human factors are broader than engineering psychology, which is focused specifically on designing systems that accommodate the information-processing capabilities of the brain.

### **Macroergonomics**

Macroergonomics is an approach to ergonomics that emphasises a broad system view of design, examining organisational environments, culture, history, and work goals. It deals with the physical design of tools and the environment. It is the study of the society/technology interface and their consequences for relationships, processes, and institutions. It also deals with the optimisation of the designs of organisational and work systems through the consideration of personnel, technological, and environmental variables and their interactions. The goal of macroergonomics is a completely efficient work system at both the macro- and micro-ergonomic level which results in improved productivity, and employee satisfaction, health, safety, and commitment. It analyses the whole system, finds how each element should be placed in the system, and considers all aspects for a fully efficient system. A misplaced element in the system can lead to total failure.

### **Seating Ergonomics**

The best way to reduce pressure in the back is to be in a standing position. However, there are times when you need to sit. When sitting, the main part of the body weight is transferred to the seat. Some weight is also transferred to the floor, backrest, and armrests. Where the weight is transferred is the key to a good seat design. When the proper areas are not supported, sitting in a seat all day can put unwanted pressure on the back causing pain.

The lumbar (bottom five vertebrae in the spine) needs to be supported to decrease disc pressure. Providing both a seat back that inclines backwards and has a lumbar support is critical to prevent excessive low back pressures. The combination which minimises pressure on the lower back is having a backrest inclination of 120 degrees and a lumbar support of 5 cm. The 120 degrees inclination means the angle between the seat and the backrest should be 120 degrees. The lumbar support of 5 cm means the chair backrest supports the lumbar by sticking out 5 cm in the lower back area.

Another key to reducing lumbar disc pressure is the use of armrests. They help by putting the force of your body not entirely on the seat and backrest, but putting some of this pressure on the armrests. Armrest needs to be adjustable in height to assure that shoulders are not overstressed.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to the fundamental concepts of Ergonomics. You have also learnt the different aspect of ergonomics and also the history of ergonomics. Ergonomics in workplace was also discussed alongside achieving efficiency in ergonomics. The various benefits of ergonomics were also discussed. The various fields of ergonomics were also briefly explained.

#### **5.0 SUMMARY**

In this unit you, have learnt the following:

- introduction to ergonomics which is derived from two Greek words: ergon, meaning work, and nomoi, meaning natural laws, to create a word that means the science of work and a person's relationship to that work
- highlighting of the various aspect of ergonomics like safety, comfort, ease of use e.t.c.
- the history of ergonomics whose foundations appears to have been laid within the context of the culture of Ancient Greece
- the discussion of ergonomics in workplace and achieving efficiency in ergonomics
- explanation of the various benefits of ergonomics which was discussed in greater detail
- the discussion of various fields of ergonomics like engineering psychology, Macroergonomics, Seating ergonomics.

#### **SELF- ASSESSMENT EXERCISE**

- i. What do you understand by ergonomics?
- ii. Highlight the various benefits of ergonomics, giving examples.

#### **6.0 TUTOR- MARKED ASSIGNMENT**

Discuss briefly any two fields of ergonomics.



## 7.0 REFERENCES/FURTHER READING

Berkeley Lab. *Integrated Safety Management: Ergonomics*. Website. Retrieved 9 July 2008.

Berkeley Lab. *Today at Berkeley Lab: Ergonomic Tips for Computer Users*. Retrieved 8 January 2009.

Brookhuis, K., Hedge, A., Hendrick, H., Salas, E., & Stanton, N. (2005). *Handbook of Human Factors and Ergonomics Models*. Florida: CRC Press.

Wickens & Hollands (2006). *Engineering Psychology and Human Performance*.

[www.wikipedia.com](http://www.wikipedia.com)

## **MODULE 4      USER INTERFACE EVALUATION**

Unit 1	Techniques for Evaluating and Measuring Interface Usability
Unit 2	Evaluating User Interface without the User
Unit 3	Evaluating User Interface with the User
Unit 4	Other Evaluation Methodologies

### **UNIT 1      TECHNIQUES FOR EVALUATING AND MEASURING INTERFACE USABILITY**

#### **CONTENTS**

1.0	Introduction
2.0	Objectives
3.0	Main Content
	3.1 Usability Evaluation Methods
	3.2 Evaluation with Tests and Metric
4.0	Conclusion
5.0	Summary
6.0	Tutor -Marked Assignment
7.0	References/Further Reading

#### **1.0 INTRODUCTION**

This unit describes usability evaluation methods. Cognitive modelling, inspection, inquiry and prototyping methods are described.

#### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain various usability evaluation methods
- explain the differences between these methods
- describe evaluation metrics.

#### **3.0 MAIN CONTENT**

##### **3.1 Usability Evaluation Methods**

There are a variety of methods currently used to evaluate usability. Certain methods make use of data gathered from users, while others rely on usability experts. There are usability evaluation methods that apply to all stages of design and development, from product definition to final

design modifications. When choosing a method you must consider the cost, time constraints, and appropriateness of the method. Usability methods can be further classified into the following subcategories:

### **3.1.1 Cognitive Modelling Methods**

Cognitive modeling involves creating a computational model to estimate how long it takes people to perform a given task. Models are based on psychological principles and experimental studies to determine times for cognitive processing and motor movements. Cognitive models can be used to improve user interfaces or predict problem errors and pitfalls during the design process. A few examples of cognitive models include:

#### **a. Parallel Design**

With parallel design, several people create an initial design from the same set of requirements. Each person works independently, and when finished, shares his/her concepts with the group. The design team considers each solution, and each designer uses the best ideas to further improve their own solution. This process helps to generate many different, diverse ideas and ensures that the best ideas from each design are integrated into the final concept. This process can be repeated several times until the team is satisfied with the final concept.

#### **b. GOMS**

GOMS is an acronym that stands for Goals, Operator, Methods, and Selection Rules. It is a family of techniques that analyses the user complexity of interactive systems. Goals are what the user has to accomplish. An operator is an action performed in service of a goal. A method is a sequence of operators that accomplish a goal. Selection rules specify which method should be used to satisfy a given goal, based on the context.

#### **c. Human Processor Model**

Sometimes it is useful to break a task down and analyse each individual aspect separately. This allows the tester to locate specific areas for improvement. To do this, it is necessary to understand how the human brain processes information. This has been fully described in module 1.

#### **d. Keystroke Level Modelling**

Keystroke level modeling is essentially a less comprehensive version of GOMS that makes simplifying assumptions in order to reduce

calculation time and complexity. You can read more about Keystroke level model for more information.

### **3.1.2 Inspection Methods**

These usability evaluation methods involve observation of users by an experimenter, or the testing and evaluation of a program by an expert reviewer. They provide more quantitative data as tasks that can be timed and recorded.

#### **a. Card Sorting**

Card sorting is a way to involve users in grouping information for a website's usability review. Participants in a card sorting session are asked to organise the content from a Web site in a way that makes sense to them. Participants review items from a Web site and then group these items into categories. Card sorting helps to learn how users think about the content and how they would organise the information on the Web site. Card sorting helps to build the structure for a Web site, decide what to put on the home page, and label the home page categories. It also helps to ensure that information is organised on the site in a way that is logical to users.

#### **b. Ethnography**

Ethnographic analysis is derived from anthropology. Field observations are taken at a site of a possible user, which track the artifacts of work such as Post-It notes, items on desktop, shortcuts, and items in trash bins. These observations also gather the sequence of work and interruptions that determine the user's typical day.

#### **c. Heuristic Evaluation**

Heuristic evaluation is a usability engineering method for finding and assessing usability problems in a user interface design as part of an iterative design process. It involves having a small set of evaluators examining the interface and using recognised usability principles (the "heuristics"). It is the most popular of the usability inspection methods, as it is quick, cheap, and easy. It is fully discussed in unit 2 of this module.

### **Usability Inspection**

Usability inspection is a review of a system based on a set of guidelines. The review is conducted by a group of experts who are deeply familiar

with the concepts of usability in design. The experts focus on a list of areas in design that have been shown to be troublesome for users.

### **i. Pluralistic Inspection**

Pluralistic inspections are meetings where users, developers, and human factors people meet together to discuss and evaluate step by step of a task scenario. As more people inspect the scenario for problems, the higher the probability to find problems. In addition, the more interaction in the team, the faster the usability issues are resolved.

### **ii. Consistency Inspection**

In consistency inspection, expert designers review products or projects to ensure consistency across multiple products to look if it does things in the same way as their own designs.

### **iii. Activity Analysis**

Activity analysis is a usability method used in preliminary stages of development to get a sense of situation. It involves an investigator observing users as they work in the field. Also referred to as user observation, it is useful for specifying user requirements and studying currently used tasks and subtasks. The data collected is qualitative and useful for defining the problem. It should be used when you wish to frame what is needed, or “What do we want to know?”

## **3.1.3 Inquiry Methods**

The following usability evaluation methods involve collecting qualitative data from users. Although the data collected is subjective, it provides valuable information on what the user wants.

### **a. Task Analysis**

Task analysis means learning about users' goals and users' ways of working. Task analysis can also mean figuring out what more specific tasks users must do to meet those goals and what steps they must take to accomplish those tasks. Along with user and task analysis, we often do a third analysis: understanding users' environments (physical, social, cultural, and technological environments).

### **b. Focus Groups**

A focus group is a focused discussion where a moderator leads a group of participants through a set of questions on a particular topic. Although

typically used as a marketing tool, focus groups are sometimes used to evaluate usability. Used in the product definition stage, a group of 6 to 10 users are gathered to discuss what they desire in a product. An experienced focus group facilitator is hired to guide the discussion to areas of interest for the developers. Focus groups are typically videotaped to help get verbatim quotes, and clips are often used to summarise opinions. The data gathered is not usually quantitative, but can help get an idea of a target group's opinion.

### **c. Questionnaires/Surveys**

Surveys have the advantages of being inexpensive, require no testing equipment, and results reflect the users' opinions. When written carefully and given to actual users who have experience with the product and knowledge of design, surveys provide useful feedback on the strong and weak areas of the usability of a design. This is a very common method and often does not appear to be a survey, but just a warranty card.

### **3.1.3 Prototyping Methods**

Rapid prototyping is a method used in early stages of development to validate and refine the usability of a system. It can be used to quickly and cheaply evaluate user-interface designs without the need for an expensive working model. This can help remove hesitation to change the design, since it is implemented before any real programming begins. One such method of rapid prototyping is paper prototyping.

### **3.1.4 Testing Methods**

These usability evaluation methods involve testing of subjects for the most quantitative data. Usually recorded on video, they provide task completion time and allow for observation of attitude.

#### **a. Remote Usability Testing**

Remote usability testing (also known as unmoderated or asynchronous usability testing) involves the use of a specially modified online survey, allowing the quantification of user testing studies by providing the ability to generate large sample sizes. Additionally, this style of user testing also provides an opportunity to segment feedback by demographic, attitudinal and behavioural type. The tests are carried out in the user's own environment (rather than labs) helping further simulate real-life scenario testing. This approach also provides a vehicle to easily solicit feedback from users in remote areas.

**b. Thinking Aloud**

The think aloud protocol is a method of gathering data that is used in both usability and psychology studies. It involves getting a user to verbalise their thought processes as they perform a task or set of tasks. Often an instructor is present to prompt the user into being more vocal as they work. Similar to the subjects-in-tandem method, it is useful in pinpointing problems and is relatively simple to set up. Additionally, it can provide insight into the user's attitude, which can not usually be discerned from a survey or questionnaire.

**c. Subjects-in-Tandem**

Subjects-in-tandem is pairing of subjects in a usability test to gather important information on the ease of use of a product. Subjects tend to think out loud and through their verbalised thoughts designers learn where the problem areas of a design are. Subjects very often provide solutions to the problem areas to make the product easier to use.

**3.1.5 Other Methods**

Cognitive walkthrough is a method of evaluating the user interaction of a working prototype or final product. It is used to evaluate the system's ease of learning. Cognitive walkthrough is useful to understand the user's thought processes and decision making when interacting with a system, specially for first-time or infrequent users.

**a. Benchmarking**

Benchmarking creates standardised test materials for a specific type of design. Four key characteristics are considered when establishing a benchmark: time to do the core task, time to fix errors, time to learn applications, and the functionality of the system. Once there is a benchmark, other designs can be compared to it to determine the usability of the system.

**b. Meta-Analysis**

Meta-Analysis is a statistical procedure to combine results across studies to integrate the findings. This phrase was coined in 1976 as a quantitative literature review. This type of evaluation is very powerful for determining the usability of a device because it combines multiple techniques to provide very accurate quantitative support.

### c. Persona

Personas are fictitious characters that are created to represent a site or product's different user types and their associated demographics and technographics. Alan Cooper introduced the concept of using personas as a part of interactive design in 1998 in his book *The Inmates Are Running the Asylum*, but had used this concept since as early as 1975.

Personas are a usability evaluation method that can be used at various design stages. The most typical time to create personas is at the beginning of designing so that designers have a tangible idea of who the users of their product will be. Personas are the archetypes that represent actual groups of users and their needs, which can be a general description of person, context, or usage scenario. This technique turns marketing data on target user population into a few physical concepts of users to create empathy among the design team, with the final aim of tailoring a product more closely to how the personas will use it.

To gather the marketing data that personas require, several tools can be used, including online surveys, web analytics, customer feedback forms, and usability tests, and interviews with customer-service representatives.

Cognitive walkthrough is fully discussed in unit 2 of this module.

## 3.2 Evaluation with Tests and Metrics

Regardless of how carefully a system is designed, all techniques must be tested using usability tests. Usability tests involve typical users using the system (or product) in a realistic environment. Observation of the user's behaviour, emotions, and difficulties while performing different tasks, often identify areas of improvement for the system.

### 3.2.1 The Use of Prototypes

It is often very difficult for designers to conduct usability tests with the exact system being designed. Cost constraints, size, and design constraints usually lead the designer to creating a prototype of the system. Instead of creating the complete final system, the designer may test different sections of the system, thus making several small models of each component of the system. The types of usability prototypes may vary from using paper models, index cards, hand drawn models, or storyboards.

Prototypes are able to be modified quickly, are often faster and easier to create with less time invested by designers and are more apt to change design; although sometimes are not an adequate representation of the



whole system, are often not durable and testing results may not be parallel to those of the actual system.

### 3.2.2 Metrics

While conducting usability tests, designers must use usability metrics to identify what it is they are going to measure, or the usability metrics. These metrics are often variable, and change in conjunction with the scope and goals of the project. The number of subjects being tested can also affect usability metrics, as it is often easier to focus on specific demographics. Qualitative design phases, such as general usability (can the task be accomplished?), and user satisfaction are also typically done with smaller groups of subjects. Using inexpensive prototypes on small user groups provides more detailed information, because of the more interactive atmosphere, and the designer's ability to focus more on the individual user.

As the designs become more complex, the testing must become more formalised. Testing equipment will become more sophisticated and testing metrics become more quantitative. With a more refined prototype, designers often test effectiveness, efficiency, and subjective satisfaction, by asking the user to complete various tasks. The tasks are measured by:

- the percentage of the task completed
- how long it takes to complete the tasks
- ratios of success to failure to complete the task
- time spent on errors
- the number of errors
- rating scale of satisfactions
- number of times user seems frustrated, etc.

Additional observations of the users give designers insight on navigation difficulties, controls, conceptual models, etc. The ultimate goal of analysing these metrics is to find/create a prototype design that users like and use to successfully perform given tasks.

After conducting usability tests, it is important for a designer to record what was observed, in addition to why such behaviour occurred and modify the model according to the results. Often it is quite difficult to distinguish the source of the design errors, and what the user did wrong. However, effective usability tests will not generate a solution to the problem, but provide modified design guidelines for continued testing.

## 4.0 CONCLUSION

Usability is now recognised as an important software quality attribute, earning its place among more traditional attributes such as performance and robustness. Indeed, various academic programs focus on usability. Also several usability consultancy companies have emerged, and traditional consultancy and design firms are offering similar services.

## 5.0 SUMMARY

In this unit, you have learnt the following:

- cognitive modelling involves a computational model to estimate how long it takes people to perform a given task
- inspection usability evaluation methods involve observation of users by an experimenter, or the testing and evaluation of a program by an expert reviewer
- inquiry usability evaluation methods involve collecting qualitative data from users. Although the data collected are subjective, they provide valuable information on what the user wants
- rapid prototyping is a method used in early stages of development to validate and refine the usability of a system. It can be used to quickly and cheaply evaluate user-interface designs without the need for an expensive working model
- testing usability evaluation methods involve testing of subjects for the most quantitative data. Usually recorded on video, they provide task completion time and allow for observation of attitude
- cognitive walkthrough is a method of evaluating the user interaction of a working prototype or final product. It is used to evaluate the system's ease of learning
- usability metrics is used to identify the features that will be measured.

## 6.0 TUTOR- MARKED ASSIGNMENT

- i. Explain cognitive modelling.
- ii. Describe GOMS briefly.
- iii. Identify some metrics and use them to evaluate the main menu interface of WINDOWS OS.

## 7.0 REFERENCES/FURTHER READING

Dumas, J. S. & Redish, J. C. (1999). *A Practical Guide to Usability Testing* (revised ed.). Bristol, U.K.: Intellect Books.

Holm, Ivar (2006). *Ideas and Beliefs in Architecture and Industrial design: How attitudes, orientations, and underlying assumptions shape the built environment*. Oslo School of Architecture and Design. [ISBN 8254701741](#).

Kuniavsky, M. (2003). *Observing the User Experience: A Practitioner's Guide to User Research*, San Francisco, CA: Morgan Kaufmann.

McKeown, Celine (2008). *Office Ergonomics: Practical Applications*. Boca Raton, FL: Taylor & Francis Group, LLC.

Wickens, C.D *et al.* (2004). *An Introduction to Human Factors Engineering* (2nd ed.). Pearson Education, Inc., Upper Saddle River, NJ: Prentice Hall.

[www.wikipedia.org](http://www.wikipedia.org)

## **UNIT 2     EVALUATING USER INTERFACE WITHOUT THE USERS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Evaluation of User Interface without the Users
    - 3.1.1 Users Walkthrough
    - 3.1.2 Action Analysis
    - 3.1.3 Heuristics Analysis
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor- Marked Assignment
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

This unit describes the concept of evaluating of user interface without the user. Users walkthrough, action analysis and heuristics analysis and various concepts that will be discussed throughout this unit.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain the concept of evaluating of user interface without the user
- describe users walkthrough
- explain action analysis
- describe heuristics analysis.

### **3.0 MAIN CONTENT**

#### **3.1 Evaluating User Interface without the Users**

Throughout this course material, we have emphasised the importance of bringing users into the interface design process. However, as a designer, you will also need to evaluate the evolving design when no users are present. Users' time is almost never a free or unlimited resource. Most users have their own work to do, and they are able to devote only limited time to your project. When users do take time to look at your design, it should be as free as possible of problems. This is a courtesy to the users, who shouldn't have to waste time on trivial bugs that you

could have caught earlier. It also helps build the users' respect for you as a professional, making it more likely that they will give the design effort serious attention.

A second reason for evaluating a design without users is that a good evaluation can catch problems that an evaluation with only a few users may not reveal. The numbers tell the story here: An interface designed for a popular personal computer might be used by thousands of people, but it may be tested with only a few dozen users before release. Every user will have a slightly different set of problems, and the testing will not uncover problems that the few users tested don't have. It also won't uncover problems that users might have after they get more experience. An evaluation without users won't uncover all the problems either. But doing both kinds of evaluation significantly improves the chances of success.

In this unit, we will describe three approaches for evaluating user interface in the absence of users. The first approach is the **cognitive walkthrough**, a task-oriented technique that fits especially well in the context of task-centred design. The second approach is **action analysis**, which allows a designer to predict the time that an expert user would need to perform a task, and which forces the designer to take a detailed look at the interface. The third approach is **heuristic evaluation**, a kind of check-list approach that catches a wide variety of problems but requires several evaluators who have some knowledge of usability problems.

### 3.1.1 Cognitive Walkthroughs

The cognitive walkthrough is a formalised way of imagining people's thoughts and actions when they use an interface for the first time. Briefly, a walkthrough goes like this: You have a prototype or a detailed design description of the interface, and you know who the users will be. You select one of the tasks that the design is intended to support. Then you try to tell a believable story about each action a user has to take to do the task. To make the story believable you have to motivate each of the user's actions, relying on the user's general knowledge and on the prompts and feedback provided by the interface. If you can't tell a believable story about an action, then you have located a problem with the interface.

You can see from the brief example that the walkthrough can uncover several kinds of problems. It can question assumptions about what the users will be thinking ("why would a user think the machine needs to be switched on?"). It can identify controls that are obvious to the design engineer but may be hidden from the user's point of view ("the user

wants to turn the machine on, but can she find the switch?"). It can suggest difficulties with labels and prompts ("the user wants to turn the machine on, but which is the power switch and which way is on?"). And it can note inadequate feedback, which may make the users balk and retrace their steps even after they've done the right thing ("how does the user know it's turned on?").

The walkthrough can also uncover shortcomings in the current specification, that is, not in the interface but in the way it is described. Perhaps the copier design really was "intended" to have a power-on indicator, but it just didn't get written into the specifications. The walkthrough will ensure that the specs are more complete. On occasion the walkthrough will also send the designer back to the users to discuss the task. Is it reasonable to expect the users to turn on the copier before they make a copy? Perhaps it should be on by default, or turn itself on when the "Copy" button is pressed.

Walkthroughs focus most on problems that users will have when they first use an interface, without training. For some systems, such as "walk-up-and-use" banking machines, this is obviously critical. But the same considerations are also important for sophisticated computer programs that users might work with for several years. Users often learn these complex programs incrementally, starting with little or no training and learning new features as they need them. If each task-oriented group of features can pass muster under the cognitive walkthrough, then the user will be able to progress smoothly from novice behaviour to productive expertise.

One other point from the example: Notice that we used some features of the task that were implicitly pulled from a detailed, situated understanding of the task: the user is sitting at a desk, so she can't see the power switch. It would be impossible to include all relevant details like this in a written specification of the task. The most successful walkthroughs will be done by designers who have been working closely with real users, so they can create a mental picture of those users in their actual environments.

Now here are some details on performing walkthroughs and interpreting their results.

#### **a. Who should do a walkthrough, and when?**

If you are designing a small piece of the interface on your own, you can do your own, informal, "in your head" walkthroughs to monitor the design as you work. Periodically, as larger parts of the interface begin to coalesce, it's useful to get together with a group of people, including other designers and users, and do a walkthrough for a complete task.

One thing to keep in mind is that the walkthrough is really a tool for developing the interface, not for validating it. You should go into a walkthrough expecting to find things that can be improved. Because of this, we recommend that group walkthroughs be done by people who are roughly at the same level in the company hierarchy. The presence of high-level managers can turn the evaluation into a show, where the political questions associated with criticising someone else's work overshadow the need to improve the design.

### **b. Preparing to do a walkthrough**

You need information about four things. (1) You need a description or a prototype of the interface. It doesn't have to be complete, but it should be fairly detailed. Things like exactly what words are in a menu can make a big difference. (2) You need a task description. The task should usually be one of the representative tasks you're using for task-centred design, or some piece of that task. (3) You need a complete, written list of the actions needed to complete the task with the interface. (4) You need an idea of who the users will be and what kind of experience they'll bring to the job. This is an understanding you should have developed through your task and user analysis.

### **c. Doing a walkthrough**

You have defined the task, the users, the interface, and the correct action sequence. You've gathered a group of designers and other interested stakeholders. Now it's time to actually DO THE WALKTHROUGH. In doing the walkthrough, you try to tell a story about why the user would select each action in the list of correct actions. And you critique the story to make sure it's believable.

We recommend keeping four questions in mind as you critique the story:

- Will users be trying to produce whatever effect the action has?
- Will users see the control (button, menu, switch, etc.) for the action?
- Once users find the control, will they recognise that it produces the effect they want?
- After the action is taken, will users understand the feedback they get, so they can go on to the next action with confidence?

Here are a few examples -- "failure stories" -- of interfaces that illustrate how the four questions apply.

The first question deals with what the user is thinking. Users often are not thinking what designers expect them to think. For example, one portable computer we have used has a slow-speed mode for its processor, to save battery power. Assume the task is to do some compute-intensive spreadsheet work on this machine, and the first action is to toggle the processor to high-speed mode. Will users be trying to do this? Answer: Very possibly not! Users don't expect computers to have slow and fast modes, so unless the machine actually prompts them to set the option, many users may leave the speed set at its default value or at whatever value it happened to get stuck in at the computer store.

The second question concerns the users' ability to locate the control, not to identify it as the right control, but simply to notice that it exists! Is this often a problem? You bet. Attractive physical packages commonly hide "ugly" controls. One of our favourite examples is an office copier that has many of its buttons hidden under a small door, which has to be pressed down so it will pop up and expose the controls. If the task is, for example, to make double-sided copies, then there's no doubt that users with some copier experience will look for the control that selects that function. The copier in question, in fact, has a clearly visible "help" sheet that tells users which button to push. But the buttons are hidden so well that many users have to ask someone who knows the copier where to find them. Other interfaces that take a hit on this question are graphic interfaces that require the user to hold some combination of keys while clicking or dragging with a mouse, and menu systems that force the users to go through several levels to find an option. Many users will never discover what they're after in these systems without some kind of training or help.

The third question involves identifying the control. Even if the users want to do the right thing and the control is plainly visible, will they realise that this is the control they're after? An early version of a popular word processor had a table function. To insert a new table the user selected a menu item named "Create Table." This was a pretty good control name. But to change the format of the table, the user had to select a menu item called "Format Cells." The designers had made an analogy to spreadsheets, but users weren't thinking of spreadsheets - they were thinking of tables. They often passed right over the correct menu item, expecting to find something called "Format Table." The problem was exacerbated by the existence of another menu item, "Edit Table," which was used to change the table's size.

Notice that the first three questions interact. Users might not want to do the right thing initially, but an obvious and well labelled control could let them know what needs to be done. A word processor, for example,



might need to have a spelling dictionary loaded before the spelling checker could run. Most users probably wouldn't think of doing this. But if a user decided to check spelling and started looking through the menus, a "load spelling dictionary" menu item could lead them to take the right action. Better yet, the "check spelling" menu item could bring up a dialog box that asked for the name of the spelling dictionary to load.

The final question asks about the feedback after the action is performed. Generally, even the simplest actions require some kind of feedback, just to show that the system "noticed" the action: a light appears when a copier button is pushed, an icon highlights when clicked in a graphical interface, etc. But at a deeper level, what the user really needs is evidence that whatever they are trying to do (that "goal" that we identified in the first question) has been done, or at least that progress has been made. Here's an example of an interface where that fails. A popular file compression program lets users pack one or more files into a much smaller file on a personal computer. The program presents a dialog box listing the files in the current directory. The user clicks on each file that should be packed into the smaller file, then, after each file, clicks the "Add" button. But there's no change visible after a file has been added. It stays in the list, and it isn't highlighted or grayed. As a result, the user isn't sure that all the files have been added, so he or she may click on some files again which causes them to be packed into the smaller file twice, taking up twice the space!

**d. What do you do with the results of the walkthrough?**

Fix the interface! Many of the fixes will be obvious: make the controls more obvious, use labels that users will recognise (not always as easy as it sounds), provide better feedback. Probably the most difficult problem to correct is one where the user doesn't have any reason to think that an action needs to be performed. A really nice solution to this problem is to eliminate the action, let the system take care of it. If that can't be done, then it may be possible to re-order the task so users will start doing something they know needs doing, and then get prompted for the action in question. The change to the "spelling dictionary" interaction that we described is one example. For the portable computer speed problem, the system might monitor processor load and ask if the user wanted to change to low speed whenever the average load was low over a 20 minute period, with a similar prompt for high speed.

### 3.1.2 Action Analysis

Action analysis is an evaluation procedure that forces you to take a close look at the sequence of actions a user has to perform to complete a task with an interface. In this unit, we will distinguish between two flavours of action analysis. The first, "formal" action analysis, is often called "keystroke-level analysis" in HCI work. The formal approach is characterised by the extreme detail of the evaluation. The detail is so fine that, in many cases, the analysis can predict the time to complete tasks within a 20 percent margin of error, even before the interface is prototyped. It may also predict how long it will take a user to learn the interface. Unfortunately, formal action analysis is not easy to do.

The second flavour of action analysis is what we call the "back of the envelope" approach. This kind of evaluation will not provide the detailed predictions of task time and interface learnability, but it can reveal large-scale problems that might otherwise get lost in the forest of details that a designer is faced with. As its name implies, the back-of-the-envelope approach does not take a lot of effort.

Action analysis, whether formal or back-of-the-envelope, has two fundamental phases. The first phase is to decide what physical and mental steps a user will perform to complete one or more tasks with the interface. The second phase is to analyse those steps, looking for problems. Problems that the analysis might reveal are that it takes too many steps to perform a simple task, or it takes too long to perform the task, or there is too much to learn about the interface. The analysis might also reveal "holes" in the interface description, things that the system should be able to do but can not. And it could be useful in writing or checking documentation, which should describe the facts and procedures that the analysis has shown the user needs to know.

#### a. Formal Action Analysis

The formal approach to action analysis has been used to make accurate predictions of the time it takes a skilled user to complete tasks. To predict task times, the times to perform each small step of the task, physical or mental, are estimated, and those times are added together. Most steps take only a fraction of a second. A typical step is a keystroke, which is why the formal approach is often called "keystroke-level analysis."

The predictions of times for each small step are found by testing hundreds of individual users, thousands of individual actions, and then calculating average values. These values have been determined for most of the common actions that users perform with computer interfaces. We

summarise those values in Table 2.1. If an interface control is not in the table, it might be possible to extrapolate a reasonable value from similar devices, or user testing might have to be done for the new control.

The procedure for developing the list of individual steps is very much like programming a computer. The basic task is divided into a few subtasks, like subroutines in a computer program. Then each of those subtasks is broken into smaller subtasks, and so on until the description reaches the level of the fraction-of-a-second operations listed in the table. The end result is a hierarchical description of a task and the action sequence needed to accomplish it.

**Table 2.1: Average Times for Computer Interface Actions**

<b>PHYSICAL MOVEMENTS</b>		
Enter one keystroke on a standard keyboard:	.28 second	Ranges from .07 second for highly skilled typists doing transcription, to .2 second for an average 60-wpm typist, to over 1 second for a bad typist. Random sequences, formulas, and commands take longer than plain text.
Use mouse to point at object on screen	1.5 second	May be slightly lower but still at least 1 second for a small screen and a menu. Increases with larger screens, smaller objects.
Move hand to pointing device or function key	.3 second	Ranges from .21 second for cursor keys to .36 second for a mouse.
<b>VISUAL PERCEPTION</b>		
Respond to a brief light	.1 second	Varies with intensity, from .05 second for a bright light to .2 second for a dim one.
Recognise a 6-letter word	.34 second	
Move eyes to new location on screen (saccade)	.23 second	
<b>MENTAL ACTIONS</b>		
Retrieve a simple item from long-term memory	1.2 second	A typical item might be a command abbreviation ("dir"). Time is roughly halved if the same item needs to be retrieved again immediately.
Learn a single "step" in a procedure	25 seconds	May be less under some circumstances, but most research shows 10 to 15 seconds as a minimum. None of these figures include the time needed to get started in a training situation.

Execute a mental "step"	.075 second	Ranges from .05 to .1 second, depending on what kind of mental step is being performed.
Choose among methods	1.2 second	Ranges from .06 to at least 1.8 seconds, depending on complexity of factors influencing the decision.

Source: Olson and Olson(1990)

Many values given in this table are averaged and rounded.

A full-blown formal analysis of a complex interface is a daunting task. The example and the size of the time values in the table should give some idea of why this is so. Imagine you want to analyse two designs for a spreadsheet to see which is faster on a given task. The task is to enter some formulas and values, and you expect it to take the skilled user on the order of 10 minutes. To apply the formal action analysis approach you'll have to break the task down into individual actions, most of which take less than a second. That comes out to around 1000 individual actions, just to analyse a single 10-minute task! (There will probably be clusters of actions that get repeated; but the effort is still nontrivial.)

A further problem with formal analysis is that different analysts may come up with different results, depending on how they see the task hierarchy and what actions they predict a user will take in a given situation. (Will the user scan left, then down the spreadsheet? Down then left? Diagonally?). The difference might be seconds, swamping other details. Questions like this may require user testing to settle.

Because it is so difficult, we think that formal action analysis is useful only in special circumstances, basically, when its high cost can be justified by a very large payoff. One instance where this was the case was the evaluation of a proposed workstation for telephone operators (see the article by Gray *et al* listed in Credits and Pointers, below). The phone company contracting the action analysis calculated that a savings of a few seconds in a procedure performed by thousands of operators over hundreds of thousands of calls would more than repay the months of effort that went into the evaluation.

Another place formal action analysis can be effective is for segments of the interface that users will access repeatedly as part of many tasks. Some examples of this are choosing from menus, selecting or moving objects in a graphics package, and moving from cell to cell within a spreadsheet. In each of these examples, a savings of a few tenths of a second in an interaction might add up to several minutes during an hour's work. This could justify a detailed analysis of competing designs.

In most cases, however, a few tenths of a second saved in performing an action sequence, and even a few minutes saved in learning it, are trivial compared to the other aspects of the interface that we emphasise in this book. Does the interface (and the system) do what the user needs, fitting smoothly into the rest of the user's work? Can the user figure out how the interface works? Does the interface's combination of control, prompt, warning messages, and other feedback allow the user to maintain a comfortable "flow" through a task? If the user makes an error, does the system allow graceful recovery? All of these factors are central, not only to productivity but also to the user's perception of the system's quality. A serious failure on any of these points is not going to be countered by shaving a few seconds off the edges of the interface.

### **b. Back-of-the-Envelope Action Analysis**

The back-of-the-envelope approach to action analysis foregoes detailed predictions in an effort to get a quick look at the big picture. We think this technique can be very valuable, and it's easy to do. Like the formal analysis, the back-of-the-envelope version has two phases: list the actions, and then think about them. The difference is that you do not need to spend a lot of time developing a detailed hierarchical breakdown of the task. You just list a fairly "natural" series of actions and work with them.

A process that works well for listing the actions is to imagine you are explaining the process to a typical user. That means you aren't going to say, "take your hand off the keyboard and move it to the mouse," or "scan down the menu to the 'chooser' item." You will probably just say, "select 'chooser' from the apple menu." You should also put in brief descriptions of mental actions, such as "remember your password," or "convert the time on your watch to 24-hour time."

Once you have the actions listed there are several questions you can ask about the interface:

- Can a simple task be done with a simple action sequence?
- Can frequent tasks be done quickly?
- How many facts and steps does the user have to learn?
- Is everything in the documentation?

You can get useful answers to all these questions without going into fraction-of-a-second details. At the action level you'd use in talking to a user, **EVERY ACTION TAKES AT LEAST TWO OR THREE SECONDS**. Selecting something from a menu with the mouse, entering a file name, deciding whether to save under a new name or the old one, remembering your directory name, watch over a user's shoulder

sometime, or videotape a few users doing random tasks, and you'll see that combined physical and mental time for any one of these actions is a couple of seconds on a good day, three or four or even ten before morning coffee. And you'll have to start measuring in minutes whenever there's any kind of an error or mistake.

By staying at this level of analysis, you're more likely to keep the task itself in mind, along with the user's work environment, instead of getting lost in a detailed comparison of techniques that essentially do the same thing. For example, you can easily use the back-of-the-envelope results to compare your system's proposed performance with the user's ability to do the same task with typewriters, calculators, and file cabinets.

This kind of action analysis is especially useful in deciding whether or not to add features to an interface, or even to a system. Interfaces have a tendency to accumulate features and controls like a magnet accumulates paperclips in a desk drawer. Something that starts out as a simple, task-oriented action sequence can very quickly become a veritable symphony of menus, dialog boxes, and keystrokes to navigate through the options that various people thought the system should offer. Often these options are intended as "time savers," but the user ends up spending an inordinate amount of time just deciding which time saver to use and which to ignore. (One message you should take away from the table of action times is that it takes real time to decide between two ways of doing something.)

A few quick calculations can give you ammunition for convincing other members of a development team which features should or should not be added. Of course, marketing arguments to the contrary may prevail: it often seems that features sell a program, whether or not they're productive. But it's also true that popular programs sometimes become so complicated that newer, simpler programs move in and take over the low end of the market. The newcomers may even eventually displace the high-functionality leaders. (An example of this on a grand scale is the effect of personal computers on the mainframe market.)

### **3.1.3 Heuristic Analysis**

Heuristics, also called guidelines, are general principles or rules of thumb that can guide design decisions. As soon as it became obvious that bad interfaces were a problem, people started proposing heuristics for interface design, ranging from short lists of very general platitudes ("be informative") to a list of over a thousand very detailed guidelines dealing with specific items such as menus, command names, and error messages. None of these efforts has been strikingly successful in

improving the design process, although they're usually effective for critiquing favourite bad examples of someone else's design. When the short lists are used during the design process, however, a lot of problems get missed; and the long lists are usually too unwieldy to apply. In addition, all heuristics require that an analyst has a fair amount of user interface knowledge to translate the general principles into the specifics of the current situation.

Recently, Jacob Nielsen and Rolf Molich have made a real breakthrough in the use of heuristics (Nielsen and Molich, 1990). They have developed a short list of general heuristics, and more importantly, they've developed and tested a procedure for using them to evaluate a design. We give the details of that procedure below, but first we want to say something about why heuristics, which are not necessarily a task-oriented evaluation technique, can be an important part of task-centred design.

The other two evaluation methods described in this unit, the cognitive walkthrough and action analysis, are task-oriented. That is, they evaluate an interface as applied to a specific task that a user would be doing with the interface. Task-oriented evaluations have some real advantages. They focus on interface problems that occur during work and they give some idea of the importance of the problems in the context of the job. Many of the problems they reveal would only be visible as part of the sequence of actions needed to complete the task. But task-oriented evaluations also have some shortcomings. The first shortcoming is coverage: There's almost never time to evaluate every task a user would perform, so some action sequences and often some controls aren't evaluated. The second shortcoming is in identifying cross-task interactions. Each task is evaluated standing alone, so task-oriented evaluations won't reveal problems such as command names or dialog-box layouts that are done one way in one task, another way in another.

Task-free evaluation methods are important for catching problems that task-oriented methods miss. Both approaches should be used as the interface develops. Now, here's how the heuristic analysis approach works.

Nielsen and Molich used their own experience to identify nine general heuristics which, as they noted, are implicit or explicit in almost all the lists of guidelines that have been suggested for HCI. Then they developed a procedure for applying their heuristics. The procedure is based on the observation that no single evaluator will find every problem with an interface, and different evaluators will often find different problems. So the procedure for heuristic analysis is this: Have

several evaluators use the nine heuristics to identify problems with the interface, analysing either a prototype or a paper description of the design. Each evaluator should do the analysis alone. Then combine the problems identified by the individual evaluators into a single list. Combining the individual results might be done by a single usability expert, but it's often useful to do this as a group activity.

### Nielsen and Molich's Nine Heuristics

- **Simple and natural dialog** - Simple means no irrelevant or rarely used information. Natural means an order that matches the task.
- **Speak the user's language** - Use words and concepts from the user's world. Don't use system-specific engineering terms. For example, it might be necessary to interact with users in the north using Hausa.
- **Minimise user memory load** - Don't make the user remember things from one action to the next. Leave information on the screen until it's not needed.
- **Be consistent** - Users should be able to learn an action sequence in one part of the system and apply it again to get similar results in other places.
- **Provide feedback** - Let users know what effect their actions have on the system.
- **Provide clearly marked exits** - If users get into part of the system that doesn't interest them, they should always be able to get out quickly without damaging anything.
- **Provide shortcuts** - Shortcuts can help experienced users avoid lengthy dialogs and informational messages that they don't need.
- **Good error messages** - Good error messages let the user know what the problem is and how to correct it.
- **Prevent errors** - Whenever you write an error message you should also ask, can this error be avoided?

The procedure works. Nielsen and Molich have shown that the combined list of interface problems includes many more problems than any single evaluator would identify, and with just a few evaluators it includes most of the major problems with the interface. Major problems, here, are problems that confuse users or cause them to make errors. The list will also include less critical problems that only slow or inconvenience the user.

How many evaluators are needed to make the analysis work? That depends on how knowledgeable the evaluators are. If the evaluators are experienced interface experts, then three to five evaluators can catch all of the "heuristically identifiable" major problems, and they can catch 75



percent of the total heuristically identifiable problems. (We'll explain what "heuristically identifiable" means in a moment.) These experts might be people who've worked in interface design and evaluation for several years, or who have several years of graduate training in the area. For evaluators who are also specialists in the specific domain of the interface (for example, graphic interfaces, or voice interfaces, or automated teller interfaces), the same results can probably be achieved with two to three evaluators. On the other hand, if the evaluators have no interface training or expertise, it might take as many as 15 of them to find 75 percent of the problems; five of these novice evaluators might find only 50 percent of the problems.

We need to caution here that when we say "all" or "75 per cent" or "50 per cent," we're talking only about "heuristically identifiable" problems. That is, problems with the interface that actually violate one of the nine heuristics. What's gained by combining several evaluators' results is an increased assurance that if a problem can be identified with the heuristics, then it will be. But there may still be problems that the heuristics themselves miss. Those problems might show up with some other evaluation method, such as user testing or a more task-oriented analysis.

Also, all the numbers are averages of past results, not promises. Your results will vary with the interface and with the evaluators. But even with these caveats, the take-home message is still very positive: Individual heuristic evaluations of an interface, performed by three to five people with some expertise in interface design, will locate a significant number of the major problems.

## **4.0 CONCLUSION**

The concept of evaluation of user interface without the users was introduced in this unit. User walkthrough, action analysis and heuristics analysis were also discussed.

## **5.0 SUMMARY**

In this unit, you have learnt the following:

- the cognitive walkthrough is a formalised way of imagining people's thoughts and actions when they use an interface for the first time
- action analysis is an evaluation procedure that forces you to take a close look at the sequence of actions a user has to perform to complete a task with an interface. In this unit, we will distinguish between two flavours of action analysis

- the formal approach to action analysis has been used to make accurate predictions of the time it takes a skilled user to complete tasks
- heuristics, also called guidelines, are general principles or rules of thumb that can guide design decisions.

## 6.0 TUTOR- MARKED ASSIGNMENT

- i. What do you understand by Nielsen and Molich's heuristics?
- ii. What are the advantages of the heuristics?

## 7.0 REFERENCES/FURTHER READING

Molich, R. & Nielsen, J. (1990). "Improving a Human-Computer Dialogue: What Designers know about Traditional Interface Design." *Communications of the ACM*, 33, pp. 338-342.

Nielsen, J. & Molich, R. (1990)"Heuristic Evaluation of User Interfaces." Proc. CHI'90 Conference on Human Factors in Computer Systems. New York: ACM, pp. 249-256.

Nielsen, J. (1992)."Usability Engineering." San Diego, CA: Academic Press.

Nielsen, J. (1992). "Finding Usability Problems through Heuristic Evaluation." Proc. CHI'92 Conference on Human Factors in Computer Systems. New York: ACM, pp. 373-380.

Olson, J. R. & Olson, G. M. (1990). "The Growth of Cognitive Modeling in Human- Computer Interaction since GOMS," *Human-Computer Interaction*, 5 pp 221-265.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). "The Cognitive Walkthrough: A Practitioner's Guide." [In Nielsen, J. & Mack, R. L. \(Eds.\), Usability Inspection Methods, New York: John Wiley & Sons.](#)

## **UNIT 3    EVALUATING THE DESIGN WITH THE USERS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Evaluating the Design with the Users
  - 3.2 Usability Testing
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

Having read through the course guide, you will have been introduced to evaluating designs with the users. The steps involved in evaluating the design with the user and usability testing are discussed in this unit.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain the steps involved in evaluating the design with the users
- highlight the significance of users presence
- explain usability testing procedure.

### **3.0 MAIN CONTENT**

#### **3.1 Evaluating with the Users**

You cannot really tell how good or bad your interface is going to be without getting people to use it. Before the whole system is finally ready, you need to do some user testing. This means having real people try to do things with the system and observing what happens. To do this, you need people, some tasks for them to perform, and some version of the system for them to work with. Let us consider these necessities in order.

##### **3.1.1 Choosing Users to Test**

The point of testing is to anticipate what will happen when real users start using your system. So the best test users will be people who are

representative of the people you expect to have as users. If the real users are supposed to be doctors, get doctors as test users. If you do not, you can be badly misled about crucial things like the right vocabulary to use for the actions your system supports.

Since it is not possible to test with all the users, there is a need to get a few representative samples for the test. This can be achieved by using various sampling techniques such as random sampling, purposive sampling, e.t.c.

If it is difficult to find really appropriate test users you may want to do some testing with people who represent some approximation to what you really want, like medical students instead of doctors, say, or maybe even premeds, or college- educated adults. This may help you get out some of the big problems (the ones you overlooked in your cognitive walkthrough because you knew too much about your design and assumed some things were obvious that aren't). But you have to be careful not to let the reactions and comments of people who aren't really the users you are targeting drive your design. Do as much testing with the right kind of test users as you can.

### **3.1.2 Getting the Users to Know what to Do**

In your test, you will be giving the test users some things to try to do, and you will be keeping track of whether they can do them. Just as good test users should be typical of real users, so test tasks should reflect what you think real tasks are going to be like. If you have been following our advice you already have some suitable tasks: the tasks you developed early on to drive your task-centred design.

You may find out that you have to modify these tasks somewhat for use in testing. They may take too long, or they may assume particular background knowledge that a random test user will not have. So you may want to simplify them. But be careful in doing this! Try to avoid any changes that make the tasks easier or that bend the tasks in the direction of what your design supports best.

If you base your test tasks on the tasks you developed for task-centred design, you'll avoid a common problem: choosing test tasks that are too fragmented. Traditional requirements lists naturally give rise to suites of test tasks that test the various requirements separately.

### **3.1.3 Providing a System for Test Users to Use**

The key to testing early in the development process is to make changes to the design without incurring big costs, is using mock-ups in the test.

The simplest mock-ups are just pictures of screens as they would appear at various stages of a user interaction. These can be drawn on paper or they can be, with a bit more work, created on the computer using a tool like HyperCard for the Mac or a similar system for Windows. A test is done by showing users the first screen and asking them what they would do to accomplish a task you have assigned. They describe their action, and you make the next screen appear, either by rummaging around in a pile of pictures on paper and holding up the right one, or by getting the computer to show the appropriate next screen.

This crude procedure can get you a lot of useful feedback from users. Can they understand what's on the screens, or are they baffled? Is the sequence of screens well-suited to the task, as you thought it would be when you did your cognitive walkthrough, or did you miss something?

To make a simple mock-up like this you have to decide what screens you are going to provide. Start by drawing the screens users would see if they did everything the best way. Then decide whether you also want to "support" some alternative paths, and how much you want to investigate error paths. Usually it won't be practical for you to provide a screen for every possible user action, right or wrong, but you will have reasonable coverage of the main lines you expect users to follow.

During testing, if users stay on the lines you expected, you just show them the screens they would see. What if they deviate, and make a move that leads to a screen you don't have? First, you record what they wanted to do: that is valuable data about a discrepancy between what you expected and what they want to do, which is why you are doing the test. Then you can tell them what they would see, and let them try to continue, or you can tell them to make another choice. You won't see as much as you would if you had the complete system for them to work with, but you will see whether the main lines of your design are sound.

Some systems have to interact too closely with the user to be well approximated by a simple mock-up. For example a drawing program has to respond to lots of little user actions, and while you might get information from a simple mock-up about whether users can figure out some aspects of the interface, like how to select a drawing tool from a palette of icons, you won't be able to test how well drawing itself is going to work. You need to make more of the system work to test what you want to test.

The thing to do here is to get the drawing functionality up early so you can do a more realistic test. You would not wait for the system to be completed, because you want test results early. So you would aim for a

prototype that has the drawing functionality in place but does not have other aspects of the system finished off.

In some cases, you can avoid implementing stuff early by faking the implementation. This is the WIZARD OF OZ method: you get a person to emulate unimplemented functions and generate the feedback users should see.

A good illustration is the work by John Gould at IBM in 1992. He tested design alternatives for a speech transcription system for which the speech recognition component was not yet ready (Nielsen, 1992). He built a prototype system in which test users' speech was piped to a fast typist, and the typist's output was routed back to the test users' screen. This idea can be adapted to many situations in which the system you are testing needs to respond to unpredictable user input, though not to interactions as dynamic as drawing.

If you are led to develop more and more elaborate approximations to the real system for testing purposes you need to think about controlling costs. Simple mock-ups are cheap, but prototypes that really work, or even Wizard of Oz setups, take substantial implementation effort.

Some of this effort can be saved if the prototype turns out to be just part of the real system. As we will discuss further when we talk about implementation, this is often possible. A system like Visual Basic or HyperCard allows an interface to be mocked up with minimal functionality but then hooked up to functional modules as they become available. So don't plan for throwaway prototypes: try instead to use an implementation scheme that allows early versions of the real interface to be used in testing.

### **3.1.4 Deciding What Data to Collect**

Now that we have people, tasks, and a system, we have to figure out what information to gather. It is useful to distinguish PROCESS DATA from BOTTOM-LINE data. Process data are observations of what the test users are doing and thinking as they work through the tasks. These observations tell us what is happening step-by-step, and, we hope, suggests WHY it is happening. Bottom-line data give us a summary of WHAT happened: how long did users take, were they successful, how many errors did they make.

It may seem that bottom-line data are what you want. If you think of testing as telling you how good your interface is, it seems that how long users are taking on the tasks, and how successful they are, is just what you want to know. We argue that process data are actually the data to

focus on first. But as a designer you will mostly be concerned with process data.

Suppose you have designed an interface for a situation in which you figure users should be able to complete a particular test task in about a half-hour. You do a test in which you focus on bottom-line data. You find that none of your test users was able to get the job done in less than an hour. You know you are in trouble, but what are you going to do about it? Now suppose instead you got detailed records of what the users actually did. You see that their whole approach to the task was mistaken, because they didn't use the screen reduction operations presented on the third screen. Now you know where your redesign effort needs to go.

We can extend this example to make a further point about the information you need as a designer. You know people weren't using frammis reduction, but do you know why? It could be that they understood perfectly well the importance of frammis reduction, but they didn't understand the screen on which these operations were presented. Or it could be that the frammis reduction screen was crystal clear but they didn't think frammis reduction was relevant.

Depending on what you decide here, you either need to fix up the frammis reduction screen, because it isn't clear, or you have a problem somewhere else. But you can't decide just from knowing that people didn't use frammis reduction.

To get the why information you really want, you need to know what users are thinking, not just what they are doing. That's the focus of the thinking-aloud method, the first testing technique we'll discuss.

### **3.1.5 Other Major Activities in Testing with Users**

The other major activities in testing with the users are:

#### **a. Giving Instructions**

The basic instructions can be very simple: "Tell me what you are thinking about as you work." People can respond easily to this, especially if you suggest a few categories of thoughts as examples: things they find confusing, decisions they are making, and the like.

There are some other points you should add. Tell the user that you are not interested in their secret thoughts but only in what they are thinking about their task. Make clear that it is the system, not the user that is being tested, so that if they have trouble it's the system's problem, not

theirs. You will also want to explain what kind of recording you will make, and how test users' privacy will be protected.

### **b. The Role of the Observer**

Even if you don't need to be available to operate a mock-up, you should plan to stay with the user during the test. You'll do two things: prompt the user to keep up the flow of comments, and provide help when necessary. But you'll need to work out a policy for prompting and helping that avoids distorting the results you get.

It's very easy to shape the comments users will give you, and what they do in the test, by asking questions and making suggestions. If someone has missed the significance of some interface feature a word from you may focus their attention right on it. Also, research shows that people will make up an answer to any question you ask, whether or not they have any basis for the answer. You are better off, therefore, collecting the comments people offer spontaneously than prodding them to tell you about things you are interested in.

Most people won't give you a good flow of comments without being pushed a bit. So say things that encourage them to talk, but that do not direct what they should say. Good choices are "Tell me what you are thinking" or "Keep talking". Bad choices would be "What do you think those prompts about frammiss mean?" or "Why did you do that?"

On helping, keep in mind that a very little help can make a huge difference in a test, and you can seriously mislead yourself about how well your interface works by just dropping in a few suggestions here and there. Try to work out in advance when you will permit yourself to help. One criterion is: help only when you won't get any more useful information if you don't, because the test user will quit or cannot possibly continue the task. If you do help, be sure to record when you helped and what you said.

A consequence of this policy is that you have to explain to your test users that you want them to tell you the questions that arise as they work, but that you won't answer them. This seems odd at first but becomes natural after a bit.

### **c. Recording**

There are plain and fancy approaches here. It is quite practical to record observations only by taking notes on a pad of paper: you write down in order what the user does and says, in summary form. But you'll find that it takes some experience to do this fast enough to keep up in real time, and that you won't be able to do it for the first few test users you see on



a given system and task. This is just because you need a general idea of where things are going to be able to keep up. A step up in technology is to make a video record of what is happening on the screen, with a **lapel mike** on the user to pick up the comments. A further step is to instrument the system to pick up a trace of user actions, and arrange for this record to be synchronised in some way with an audio record of the comments. The advantage of this approach is that it gives you a machine readable record of user actions that can be easier to summarise and access than video.

A good approach to start with is to combine a video record with written notes. You may find that you are able to dispense with the video, or you may find that you really want a fancier record. You can adapt your approach accordingly. But if you do not have a video setup, do not let that keep you from trying the method.

#### **d. Summarising the Data**

The point of the test is to get information that can guide the design. To do this you will want to make a list of all difficulties users encountered. Include references back to the original data so you can look at the specifics if questions arise. Also try to judge why each difficulty occurred, if the data permit a guess about that.

#### **e. Using the Results**

Now you want to consider what changes you need to make to your design based on data from the tests. Look at your data from two points of view. First, what do the data tell you about how you **THOUGHT** the interface would work? Are the results consistent with your cognitive walkthrough or are they telling you that you are missing something? For example, did test users take the approaches you expected, or were they working a different way? Try to update your analysis of the tasks and how the system should support them based on what you see in the data. Then use this improved analysis to rethink your design to make it better support what users are doing.

Second, look at all of the errors and difficulties you saw. For each one make a judgement of how important it is and how difficult it would be to fix. Factors to consider in judging importance are the costs of the problem to users (in time, aggravation, and possible wrong results) and what proportion of users you can expect to have similar trouble. Difficulty of fixes will depend on how sweeping the changes required by the fix are: changing the wording of a prompt will be easy, changing the organisation of options in a menu structure will be a bit harder, and

so on. Now decide to fix all the important problems, and all the easy ones.

### **3.1.6 Measuring Bottom-Line Usability**

We argue that you usually want process data, not bottom-line data, but there are some situations in which bottom-line numbers are useful. You may have a definite requirement that people must be able to complete a task in a certain amount of time, or you may want to compare two design alternatives on the basis of how quickly people can work or how many errors they commit. The basic idea in these cases is that you will have people perform test tasks, you will measure how long they take and you will count their errors.

Your first thought may be to combine this with thinking-aloud test: in addition to collecting comments you'd collect these other data as well. Unfortunately this doesn't work as well as one would wish. The thinking-aloud process can affect how quickly and accurately people work. It's pretty easy to see how thinking-aloud could slow people down, but it has also been shown that sometimes it can speed people up, apparently by making them think more carefully about what they are doing, and hence helping them choose better ways to do things. So if you are serious about finding out how long people will take to do things with your design, or how many problems they will encounter, you really need to do a separate test.

Getting the bottom-line numbers won't be too difficult. You can use a stopwatch for timings, or you can instrument your system to record when people start and stop work. Counting errors, and gauging success on tasks, is a bit trickier, because you have to decide what an error is and what counts as successful completion of a task. But you won't have much trouble here either as long as you understand that you can't come up with perfect criteria for these things and use your common sense.

#### **a. Analysing the Bottom-Line Numbers**

When you've got your numbers you'll run into some difficult problems. The trouble is that the numbers you get from different test users will be different. How do you combine these numbers to get a reliable picture of what is happening?

#### **Illustration 1:**

Suppose users need to be able to perform some task with your system in 30 minutes or less. You run six test users and get the following times:

- 20 min
- 15 min
- 40 min
- 90 min
- 10 min
- 5 min

Are these results encouraging or not? If you take the average of these numbers you get 30 minutes, which looks fine. If you take the **MEDIAN**, that is, the middle score, you get something between 15 and 20 minutes, which look even better. Can you be confident that the typical user will meet your 30-minute target?

The answer is no. The numbers you have are so variable, that is, they differ so much among themselves, that you really can't tell much about what will be "typical" times in the long run. Statistical analysis, which is the method for extracting facts from a background of variation, indicates that the "typical" times for this system might very well be anywhere from about 5 minutes to about 55 minutes. Note that this is a range for the "typical" value, not the range of possible times for individual users. That is, it is perfectly plausible given the test data that if we measured lots and lots of users the average time might be as low as 5 min, which would be wonderful, but it could also be as high as 55 minutes, which is terrible.

There are two things contributing to our uncertainty in interpreting these test results. One is the small number of test users. It's pretty intuitive that the more test users we measure the better an estimate we can make of typical times. Second, as already mentioned, these test results are very variable: there are some small numbers but also some big numbers in the group. If all six measurements had come in right at (say) 25 minutes, we could be pretty confident that our typical times would be right around there. As things are, we have to worry that if we look at more users we might get a lot more 90-minute times, or a lot more 5-minute times.

It is the job of statistical analysis to juggle these factors: the number of people we test and how variable or consistent the results are and give us an estimate of what we can conclude from the data. This is a big topic, and we won't try to do more than give you some basic methods and a little intuition here.

Here is a statistical procedure (Computation of standard deviation and standard error) for getting an idea of the range of typical values that are consistent with your test data.

- Add up the numbers. Call this result "sum of x". In our example this is 180.
- Divide by the n, the number of numbers. The quotient is the average, or mean, of the measurements. In our example this is 30.
- Add up the squares of the numbers. Call this result "sum of squares" In our example this is 10450.
- Square the sum of x and divide by n. Call this "foo". In our example this is 5400.
- Subtract foo from sum of squares and divide by n-1. In our example this is 1010.
- Take the square root. The result is the "standard deviation" of the sample. It is a measure of how variable the numbers are. In our example this is 31.78, or about 32. 7.
- Divide the standard deviation by the square root of n. This is the "standard error of the mean" and is a measure of how much variation you can expect in the typical value. In our example this is 12.97, or about 13.

Instead of following the steps above, you may simply use formulas for computing standard deviation and standard error.

It is plausible that the typical value is as small as the mean minus two times the standard error of the mean, or as large as the mean plus two times the standard error of the mean. In our example this range is from  $30 - (2 * 13)$  to  $30 + (2 * 13)$ , or about 5 to 55. (The "\*" stands for multiplication.)

What does "plausible" mean here? It means that if the real typical value is outside this range, you were very unlucky in getting the sample that you did. More specifically, if the true typical value were outside this range you would only expect to get a sample like the one you got 5 percent of the time or less.

Experience shows that usability test data are quite variable, which means that you need a lot of it to get good estimates of typical values. If you pore over the above procedure enough you may see that if you run four times as many test users you can narrow your range of estimates by a factor of two: the breadth of the range of estimates depends on the square root of the number of test users. That means a lot of test users to get a narrow range, if your data are as variable as they often are.

What this means is that you can anticipate trouble if you are trying to manage your project using these test data. Do the test results show we are on target, or do we need to pour on more resources? It's hard to say. One approach is to get people to agree to try to manage on the basis of the numbers in the sample themselves, without trying to use statistics to

figure out how uncertain they are. This is a kind of blind compromise: on the average the typical value is equally likely to be bigger than the mean of your sample, or smaller. But if the stakes are high, and you really need to know where you stand, you'll need to do a lot of testing. You'll also want to do an analysis that takes into account the cost to you of being wrong about the typical value, by how much, so you can decide how big a test is really reasonable.

## **b. Comparing Two Design Alternatives**

If you are using bottom-line measurements to compare two design alternatives, your ability to draw a firm conclusion will depend on how different your numbers are, as well as how many test users you use. But then you need some way to compare the numbers you get for one design with the numbers from the others.

### **i. Between-Groups Experiment**

The simplest approach to use is called a BETWEEN-GROUPS EXPERIMENT. You use two groups of test users, one of which uses version A of the system and the other version B. What you want to know is whether the typical value for version A is likely to differ from the typical value for version B, and by how much. Here's a cookbook procedure for this.

Using parts of the cookbook method above, compute the means for the two groups separately (Say,  $m_a$  and  $m_b$ ). Also compute their standard deviations (Say,  $s_a$ ,  $s_b$ ). You'll also need to have  $n_a$  and  $n_b$ , the number of test users in each group (usually you'll try to make these the same, but they don't have to be.)

Combine  $s_a$  and  $s_b$  to get an estimate of how variable the whole scene is, by computing

$$s = \sqrt{ ( n_a*(s_a**2) + n_b*(s_b**2) ) / (n_a + n_b - 2) }$$

("\*" represents multiplication; "sa\*\*2" means "sa squared").

Compute a combined standard error:

$$se = s * \sqrt{1/n_a + 1/n_b}$$

The difference between versions or your range of typical values for the difference between version A and version B is now:

$$(m_a - m_b) \pm (2*se)$$

## ii. Within-Groups Experiment

Another approach you might consider is a WITHIN-GROUPS EXPERIMENT. Here you use only one group of test users and you get each of them to use both versions of the system. This brings with it some headaches. You obviously can't use the same tasks for the two versions, since doing a task the second time would be different from doing it the first time, and you have to worry about who uses which system first, because there might be some advantage or disadvantage in being the system someone tries first. There are ways around these problems, but they aren't simple. They work best for very simple tasks about which there are not much to learn. You might want to use this approach if you were comparing two low-level interaction techniques, for example. You can learn more about the within-groups approach from any standard text on experimental psychology.

## 3.2 Usability Testing

### 3.2.1 Introduction to Usability Testing

I have noticed that the term usability testing is often used rather indiscriminately to refer to any technique used to evaluate a product or system. Throughout this material, the term usability testing is referred to as the process that employs participants who are representative of the target population to evaluate the degree to which a product (User interface) meets specific usability criteria. This inclusion of representative users eliminates labelling as usability testing such techniques as expert evaluations, walkthrough, and like that do not require representative users as part of the process.

Usability testing is a research tool, with its roots in classical experimental methodology. The range of tests one can conduct is considerable, from true classical experiments with large sample sizes and complex test designs, to very informal qualitative studies with only a single participant. Each testing approach has different objectives, as well as different time and resource requirements. The emphasis of this book will be on more informal, less complex tests designed for quick turnaround of results in industrial product development environments.

### 3.2.2 Preparing for Usability Testing

For many of those contemplating the implementation of the usability testing program, the discipline has become synonymous with a high-powered, well appointed, well equipped, expensive laboratory. For some organisations, the usability lab (and by that I mean physical plant) has become more prominent and more important than the testing

process itself. Some organisations, in their zeal to impress customers and competitors alike with their commitment to usability, have created awe-inspiring palaces of high-tech wizardry prior to laying the foundation for an on-going testing program. Not realising that instituting a program of usability engineering requires a significant shift in the culture of the organisation, these organisations have put the proverbial cart before the horse in their attempt to create instant programs, rather than building programs over time.

This approach to usability testing is rather superficial and short-sighted, and has a high risk of failure. It approaches usability engineering as the latest fad to be embraced rather than as a program that requires effort, commitment, and time in order to have lasting effects on the organisation and its products. I know of at least two organisations with newly built, sophisticated usability laboratories that unfortunately are now operating as the world's most elaborate storage rooms. (An analogy is a retail store that requires and outfits a new store for business, only to realise that it does not have any interested customers). Having succumbed to the misperception that equates the laboratory with the process itself, these organisations have discovered only too late that usability testing is much more than a collection of cameras and recorders. Rather, a commitment to usability must be embedded in the very philosophy and underpinning of the organisation itself in order to guarantee success.

In that vein, if you have been charged with developing a testing program and have been funded to build an elaborate testing lab as the initial step, resist the temptation to accept the offer. Rather, start small and build the organisation from the ground up instead of from top down.

Regardless of whether you will be initiating a large testing program or simply testing your own product, you need not have elaborate, expensive lab to achieve your goals.

### **3.2.3 Six Stages of Conducting a Usability Test**

#### **a. Developing the Test Plan**

The test plan is the foundation for the entire test. It addresses the how, when, where, who, why and what of your usability test. Under the sometimes unrelenting time pressure of project deadline, there could be a tendency to forego writing a detailed test plan. Perhaps, feeling that you have a good idea of what you would like to test in your head, you decide not to bother writing it down. This informal approach is a mistake, and invariably will come to haunt you.

The following are some important reasons it is necessary to develop a comprehensive test plan, as well as some ways to use it as a communication vehicle among the development team.

**It serves as the blueprint for the test.** Much as the blueprint for a house describes exactly what you will build, the test plan describes exactly how you will go about testing your product. Just as you would not want your building contractor to “wing it” when building your house, so the exact same logic applies here. The test plan sets the stage for all that will follow. You do not want to have any loose ends just as you are about to test your first participant.

**It serves as the main communication vehicle among the main developer, the test monitor, and the rest of the development team.** The test plan is the document that all involved member of the development team as well as management (if it is interested and involved) should review in order to understand how the test will proceed and see whether their particular needs are being met. Use it to get buy-in and feedback from other members to ensure that everyone agrees on what will transpire. Since projects are dynamic and change from day to day and from week to week, you do not want someone to say at the end of the test that his or her particular agenda was not addressed. Especially when your organisation is first starting to test, everyone who is directly affected by the test results should review the test plan. This makes good business sense and political sense too.

**It describes or implies required resources, both internal and external.** Once you delineate exactly what will happen and when, it is a much easier task to foretell what you will need to accomplish your test. Either directly or by implication, the test plan should communicate the resources that are required to complete the test successfully.

**It provides a real focal point for the test and a milestone for the product being tested.** Without the test plan, details get fuzzy and ambiguous, especially under time pressure. The test plan forces you to approach the job of testing systematically, and it reminds the development team of the impending dates. Having said all that, it is perfectly acceptable and highly probable that the test plan will be developed in stages as you gradually understand more of the test objectives and talk to the people who will be involved. Projects are dynamic and the best laid plans will change as you begin to approach testing. By developing the test plan in stages, you can accommodate changes.

For example, as your time and resource constraints become clearer, your test may become less ambitious and simpler. Or, perhaps you will not be



able to acquire as many qualified participants as you thought. Perhaps not all modules or section of the document will be ready in time. Perhaps your test objectives are too imprecise and need to be simplified and focused. These are all real-world example that force you to revise the test and test plan.

A sound approach is to start writing the test plan as soon as you know you will be testing. Then, as the project proceeds, continue to refine it, get feedback, buy-in, and so forth. Of course, there is a limit to flexibility, so you need to set a reasonable deadline prior to the test after which the test plan may not change. Let that date serve as a point at which the product can no longer change until after the test. You may find that the test is the only concrete milestone at that point in time in the development cycle and, as such, serves an important function.

Once you reach the cut-off date, do all that you can to freeze the design of the product you will have to test. Additional revisions may invalidate the test design you have chosen, the question ask, even the way you collect data. If you are pressured to revise the test after the cut-off date, make sure everyone understand the risks involved. The test may be invalidated, and the product may not work properly with changes made so close to the test date.

Remember to keep the end user in mind as you develop the test plan. If you are very close to the project, there is a tendency to forget that you are not testing the product you are testing its relationship to a human being with certain specific characteristics.

### **Suggested Format**

Test plan formats will vary according to the type of test and the degree of formality required in your organisation. However, following are the typical sections to include:

- Purpose
- Problem statement/test objectives
- User profile
- Method (test design )
- Task list
- Test environment/equipment
- Test monitor role
- Evaluation measures (data to be collected)
- Report contents and presentation

## **b. Selecting and Acquiring Participants**

The selection and acquisition of participant whose background and abilities are representative of your product's intended end user is a crucial element of the testing process. After all, your test result will only be valid if the people you test are typical end users of the product, or as close to that criterion as possible. If you test the “wrong” people, it does not matter how much effort you put into the rest of the test preparation. Your result will be questionable and of limited value.

Selecting participants involves identifying and describing relevant skills and knowledge of the person(s) who will use your product. This description is known as user profile or user characterisation of the target population and should have been developed in the early stages of the product development. Then, once that has been determined, you must ascertain the most effective way to acquire people from this target population to serve as participants within your constraint of time, money, resources, and so on.

## **c. Preparing the Test Materials**

One of the more labour-intensive activities required to conduct a usability test is developing the test material that will be used to communicate with the participants, collect the data, and satisfy legal requirements. It is important to develop all important test materials well in advance of the time you will need them. Apart from the obvious benefits of not having to scurry around at the last minute, developing materials early on time helps to explicitly structure and organise the test. In fact, if you have difficulty developing one particular type of test material, it can be a sign that there are flaws in your test objectives and test design.

While the specific content of the material will vary from test to test, the general categories required will hardly vary at all. This unit contains a list of the most common materials you need to develop a test, as well as examples of the various types of test materials. As you develop them, think of these materials as aids to the testing process. Once they are developed, their natural flow will guide the test for you. Be sure to leave enough time to include the materials in your pilot test. The test materials reviewed are as follows:

- Screening questionnaire
- Orientation script
- Background questionnaire
- Data collection instruments (data loggers)
- Nondisclosure agreement and tape consent form

- Pre-test questionnaire
- Task scenarios
- Prerequisite training materials
- Post-test questionnaire
- Debriefing topics guide

#### **d. Conducting the Test**

Having completed the basic groundwork and preparation for your usability test, you are almost ready to begin testing. While there exists an almost endless variety of sophisticated esoteric tests one might conduct (from a test comprising a single participant and lasting several days to a fully automated test with 200 or more participants). It is necessary to focus on the guidelines and activities for performing classic “one-to-one” test. This “typical” test consists of four to ten participants, each of whom is observed and questioned individually by a test monitor seating in the same room. This method will work for any of the four types of tests mentioned: exploratory, assessment, validation, or comparison. The main difference is the type of objectives pursued, that is, more conceptual for an exploratory test, more behaviour oriented for assessment and validation tests. The other major difference is the amount of interaction between the participant and the test monitor. The earlier explanatory test will have much interaction. The later validation test will have much less interaction, since the objective is measurement against standard.

For “first time” testers, I recommend beginning with an assessment test; it is probably the most straightforward to conduct.

It is important to test the whole integrated product and not just separate components. Testing a component, such as documentation, separately, without ever testing it with the rest of the product, does nothing to ensure ultimate product usability. Rather it enforces the lack of product integration. In short, you eventually would like to test all components together with enough lead time to make revisions as required. However, that being said, there is absolutely nothing wrong with testing separate components as they are developed throughout the life cycle, as long as you eventually test them all together.

There is one exception to this rule, if you believe that the only way to begin any kind of testing program within an organisation is to test a component separately as your only test, then by all means do so. However, you should explain to management the limited nature of those results.

**e. Debriefing the Participant**

Debriefing refers to the interrogation and review with the participant of his or her own actions during the performance portion of a usability test. After all, one could argue that debriefing is really an extension of the testing process. Participants perform some tasks, and you interrogate that either in phases or after the entire test.

But the more I thought about how much I had learned during the debriefing portions of tests, the more I felt debriefing warranted its own separate treatment, or stage of testing. More often than not, the debriefing session is the key to understanding how to fix the problems uncovered during the performance portion of the test. While the performance of the usability test uncovers and exposes the problems, it is often the debriefing session that shed light on why these problems have occurred. Quite often, it is not until the debriefing session that one understands motive, rationale, and very subtle points of confusion. If you think of usability test as a mystery to be solved, it is not until the debriefing session that all pieces come together.

**f. Transforming Data into Findings and Recommendations**

Finally, you have completed testing and are now ready to dive in and transform a wealth of data into recommendations for improvement. Typically, the analysis of data falls into two distinct processes with two different deliverables.

The first process is a preliminary analysis and is intended to quickly ascertain the hot spots (i.e., word problems), so that the designers can work on these immediately without having to wait for the final test report. This preliminary analysis takes place as soon as it is feasible after testing has been completed. Its deliverable is either a small written report or a verbal presentation of findings and recommendation.

The second process is a comprehensive analysis, which takes place during a two-to-four-week period after the test. Its deliverables is a final, more exhaustive report. This final report should include all the findings in the preliminary report, updated if necessary, plus all other analysis and findings that were not previously covered.

A word of caution is in order regarding preliminary findings and recommendations. Developing and reporting preliminary recommendations creates a predicament for the test monitor. Your recommendations must be timely so that members of the development team, such as designers and writers, can begin implementing changes. However, you also need to be thorough, in the sense of not missing

anything important. Once preliminary recommendations are circulated for public consumption, they quickly lose their preliminary flavour. Designers will begin to implement changes, and it is difficult to revisit changes at a later time and say, “Oops”, I don’t think we should change that module after all.

You could simply avoid producing preliminary recommendations, but designers viewing the test are sure to act on what they see prior to your final report, therefore not providing preliminary recommendation is not a satisfactory option. The best compromise is to provide preliminary findings and recommendations, but be cautious and err on the conservative side by providing too little rather than too much. Stick to very obvious problems that do not require further analysis on your part. If you are unsure about a finding or a recommended solution without performing further analysis, simply say so.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to methods of testing the design with users and the stages involved in carrying out usability testing.

#### **5.0 SUMMARY**

In this unit, you have learnt the following:

- evaluating with users involves having real users to do things with the system and observing what happens
- the best test users will be people who are representative of the people you expect to have as users
- the test tasks should reflect what you think real tasks are going to be like
- choosing a user also involves giving out instructions, getting an observer, recording, summarising the data and using the result
- setting up usability study includes choosing the order of the test tasks, training test users, the pilot study, e.t.c.
- The stages involved in usability testing are:
  - develop the test plan
  - selecting and acquiring participants
  - preparing test materials
  - conducting the test
  - debriefing the participants
  - transforming data into findings and recommendations.

## 6.0 TUTOR -MARKED ASSIGNMENT

- i. Explain bottom-line numbers.
- ii. Describe a mock-up.
- iii. How would you select test users who are true representation of the users' population?
- iv. What are the goals of usability testing?
- v. Justify whether the usability testing steps described in this unit are adequate.

## 7.0 REFERENCES /FURTHER READING

Mayhew, D. (1999). *The Usability Engineering Lifecycle*. San Francisco, CA: Morgan Kaufman.

Molich, R. & Nielsen, J.(1990). "Improving a Human-Computer Dialogue: What Designers know about Traditional Interface Design." *Communications of the ACM*, 33, pp. 338-342.

Nielsen, J. (1992). "Finding Usability Problems through Heuristic Evaluation." Proc. CHI'92 Conference on Human Factors in Computer Systems. New York: ACM, pp. 373-380.

Nielsen, J. (1992)."Usability Engineering." San Diego, CA: Academic Press.

Nielsen, J. & Molich, R. (1990). "Heuristic Evaluation of User Interfaces." Proc. CHI'90 Conference on Human Factors in Computer Systems. New York: ACM, pp. 249-256.

Rubin, J. (1994). "Handbook of Usability Testing", John Wiley & Sons Inc.

## **UNIT 4 OTHER EVALUATION ISSUES**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Content
  - 3.1 Other Modelling Techniques
  - 3.2 Current Issues Concerning Evaluation Methodologies
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor -Marked Assignment
- 7.0 Reference/Further Reading

### **1.0 INTRODUCTION**

This unit describes other evaluation issues. Advantages and disadvantages of model-based evaluation techniques are discussed. Current issues concerning evaluation methodologies are also mentioned.

### **2.0 OBJECTIVES**

At the end of this unit, you should be able to:

- explain new modeling techniques
- highlight the advantages and disadvantages of model-based evaluations
- discuss current issues concerning evaluation methodologies.

### **3.0 MAIN CONTENT**

#### **3.1 Modelling Techniques**

The EPIC (Executive-Process/Interactive Control) system simulates the human perceptual and motor performance system. EPIC can interact as a human would with a simulation of a user interface system. EPIC is being used to study users engaged in multiple tasks, such as using a car navigation system while driving. Using EPIC involves writing production rules for using the interface and writing a task environment to simulate the behaviour of the user interface.

A model of information foraging useful in evaluating information seeking in web sites is based on the Active Control of Thought – Rational (ACT-R) model. The ACT-R model was developed to use in testing simulated users interacting with designs for web sites and

predicts optimal behaviour in large collections of web documents. The information foraging model is being used to understand the decisions that users of the web make in following various links to satisfy information goals.

### **3.1.1 Advantages and Disadvantages of Model-Based Evaluations**

The use of models to predict user behaviour is less expensive than empirical, user-centred evaluations. Thus many more iterations of the design can be tested. However, a necessary first step is conducting the cognitive task analysis that is needed in producing model description. This is time consuming but can be used for testing many user interface designs.

Models must be tested for validity. This is accomplished by watching humans perform the tasks and coding their behaviour for comparison with the model. This is time consuming but necessary to determine if what the model will predict is accurate.

## **3.2 Other Evaluation Techniques**

The other similar evaluation techniques are:

### **a. Formative Evaluations**

Formative evaluations obtain user feedback for early concepts or designs of software products. Formative evaluations are typically more informal in that the goal is to collect information to be used for design as opposed to collecting measures of usability. The primary source of data in formative evaluations is verbal data from the user. Early evaluations may use paper prototypes or initial screen designs. Later evaluations can be done on partial prototypes or prototypes of only one portion of a user interface. When possible, logging software is also used to capture user interaction with the software. Additionally, usability engineers often take notes of critical incidents that occur during the evaluation. The debriefing or post-evaluation interview is an excellent source of information in formative evaluations. Usability engineers can probe in depth to understand sources of confusion in the interface.

Formative evaluations need to be conducted in a fairly rapid pace in order to provide design input when it is needed. As a consequence, the evaluations usually focus on a small portion of the user interface, involve relatively few user-participants, and have less formal reporting mechanisms than summative evaluations. Ideally, software designers



and developers can observe the evaluations and discuss results and potential solutions with the usability engineers after the evaluations.

### **b. Summative Evaluations**

Summative evaluations are more formal evaluations conducted to document the usability characteristics of a software product. These evaluations involve a number of users. The recommendation is five to seven users per cell, where a cell represents a class of end-users. For example, if a product is being design for both home and small business users, then representatives of users of each type must be included in the evaluation. If both adults and teenagers will be using the home product, then representatives from both groups need to be included as evaluation participants. Good experimental design is essential to summative evaluation. The metrics of efficiency, effectiveness, and user satisfaction are typically used and the design of the evaluation must include the measures and collection methodology. Tasks used in the evaluation usually represent core functionality of the software but may also include new or improved functionality. Directions and materials given to the users need to be designed and tested in a pilot evaluation to make sure that they are understandable. Usability evaluation has always tried to make the “context-of-use” as realistic as possible. However, a usability laboratory cannot duplicate actual conditions of use within an office or home. Interruptions and other demands for attention do not, and cannot, occur during usability evaluation conditions. As such these evaluations represent the best case condition. If the software is not usable in the laboratory, it will certainly not be usable in real-world use. However, usability in the laboratory does not guarantee usability in more realistic conditions.

The desired level of usability should be defined early in the usability engineering lifecycle and the actual results from the summative evaluation are compared to this. If good usability engineering practices have been followed, including a number of formative evaluations, it is likely that the desired levels will be achieved. If this is not the case, then a decision must be made as to whether or not to release the software or to redesign and re-evaluate the usability.

### **c. Expert-based Evaluations**

Expert evaluations of usability are similar to design reviews of software projects and code walkthroughs. Inspection methods include heuristic evaluation, guideline reviews, pluralistic walkthroughs, consistency inspections, standards inspections, cognitive walkthroughs, formal usability inspections, and feature inspections.

### 3.3 Current Issues Concerning Evaluation Methodologies

While the human computer interaction community has come a long way in developing and using methods to evaluate usability, the problem is by no means solved. Although a numbers of studies have been done to compare these methods, the comparison is difficult and flaws have been pointed out in a number of these studies. First, there is the issue of using experimental (user-centred) methods to obtain answers to large questions of usability as opposed to the more narrow questions that are the more traditional use for experimental methods. A second issue is what should be used for the comparison? Should user-centred methods be considered as the ground truth? All usability tests are not created equal. There are certainly flaws in the way tests are design, conducted, and analysed. While individual methods have limitations and can be flawed in their implementation, it is certain that performing some evaluation methodology is better than doing nothing. The current best practice is to use a number of different evaluation methodologies to provide rich data on usability.

Evaluation methodologies were, for the most part, developed to evaluate the usability of desk-top systems. The current focus in technology development of mobile and ubiquitous computing presents challenges for current usability evaluation methods. Laboratory evaluations will be hard pressed to simulate use conditions for these applications. Going out into the field to evaluate use places constraints on how early evaluations can be done. Mobile and multi-user systems must be evaluated for privacy and any usability issues entailed in setting up, configuring, and using such policies. The use of such devices in the context of doing other work also has implications for determining the context of use for usability testing. We need to test car navigation systems in the car – not the usability lab.

Technology is being used by more users. The range of users using mobile phones, for example, means that representative users need to be selected from teenagers to grandparents. The accessibility laws in the United States require that federal information is accessible by persons with disabilities. Again, this requires inclusion of more users from the disable population in user-centred evaluations.

Web sites are also of interest in usability evaluation. Again, there is a matter of a broad user population. Design and development cycles in web site development are extremely fast and doing extensive usability evaluation is usually not feasible. Usability practitioners are looking at remote testing methods to more closely replicate context of usage for web site evaluation.

International standards exist for user-centred design processes, documentation, and user interfaces. Usability is becoming a requirement for companies in purchasing software as they recognise that unusable software will increase the total cost of ownership.

New usability evaluation methodologies will be developed to meet the demands of our technology-focused society. Researchers and practitioners in usability will need to join forces to meet this challenge.

#### **4.0 CONCLUSION**

In this unit, you have been introduced to evaluation issues. Advantages and disadvantages of model-based were mentioned, as well as current issues in evaluation methodologies.

#### **5.0 SUMMARY**

In this unit, you have learnt the following:

- the EPIC (Executive-Process/Interactive Control) system simulates the human perceptual and motor performance system
- the ACT-IF model was developed to use in testing simulated users interacting with designs for web sites and predicts optimal behaviour in large collections of web documents
- other evaluation techniques are formative, summative and expert system
- the use of models to predict user behaviour is less expensive than empirical, user-centred evaluations
- the main disadvantages of model-based methodologies is that it is time consuming.

#### **6.0 TUTOR -MARKED ASSIGNMENT**

What is the significance of EPIC (Executive-Process/Interactive Control) system?

#### **7.0 REFERENCE/FURTHER READING**

Mayhew, D. (1999). *The Usability Engineering Lifecycle*. San Francisco, CA: Morgan Kaufman.